

版权注意事项：

- 1、书籍版权归作者和出版社所有
- 2、本PDF仅限用于个人获取知识，进行私底下的知识交流
- 3、PDF获得者不得在互联网上以任何目的进行传播
- 4、如觉得书籍内容很赞，请购买正版实体书，支持作者
- 5、请于下载PDF后24小时内删除本PDF。

本书从入门到高级应用，从Zabbix基本应用到安装包定制、协议、API，全面剖析、应有尽有

包含大量监控案例，详解触发器、告警等Zabbix监控中令人头疼的问题，学到的不止是技术，还有思路和方法

企业级开源监控系统必选

Zabbix

企业级分布式监控系统

吴兆松 编著

本书附赠超值51CTO学院
课程学习卡**100金币**，使用
时请取出卡片。

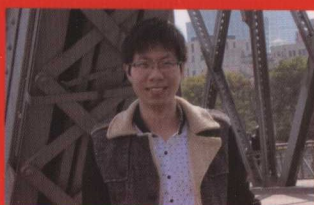


中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

作者简介



吴兆松

运维开发工程师，具有丰富的一线运维实战经验，曾做过CDN运维、游戏运维、云计算运维，对Linux系统、云计算、监控系统有较深入的研究，熟悉运维自动化工具的使用和二次定制开发，乐于折腾开源软件，偶尔写写博客、技术文档，他始终坚信，技术的积累只是时间问题，而解决问题的思路 and 思想高于具体的技术细节。

Zabbix

企业级分布式监控系统

吴兆松 编著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书从运维（OPS）角度对 Zabbix 的各项功能进行了详细介绍，以自动化运维视角为出发点，对 Zabbix 的安装配置、自动化功能、监报告警、性能调优、Zabbix API、Zabbix 协议、RPM 安装包定制，结合 saltstack 实现自动化配置管理等内容进行了全方位的深入剖析。

全书分为初级内容、中级内容、高级内容和附录部分，分别由浅入深地讲解 Zabbix 监控系统这个开源软件。从最简单的安装配置，到复杂的高级使用，详细讲解了数据库分表、高可用、性能调优、架构设计，以及大量的监控案例，对即将构建 Zabbix 监控系统，或者已经在使用 Zabbix 的用户具有非常高的参考价值。

本书适合想了解、学习和规划构建监控系统的人员阅读，可以作为学习 Zabbix 的工具书，也适合有一定基础，想更深入学习 Zabbix 的读者，通过大量的案例，让读者真正理解 Zabbix。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目（CIP）数据

Zabbix 企业级分布式监控系统 / 吴兆松编著. —北京：电子工业出版社，2014.8
ISBN 978-7-121-23877-2

I. ①Z… II. ①吴… III. ①计算机监控系统 IV. ①TP277

中国版本图书馆 CIP 数据核字（2014）第 169485 号

责任编辑：李利健

印 刷：北京季峰印刷有限公司

装 订：北京季峰印刷有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编：100036

开 本：720×1000 1/16 印张：22.25 字数：440 千字

版 次：2014 年 8 月第 1 版

印 次：2016 年 9 月第 5 次印刷

定 价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）51260888-819，faq@phei.com.cn。

前言

为何要写这本书

“运筹帷幄之中，决胜千里之外。”在 IT 运维中，监控占据着重要的地位，按比例来算，说 30% 一点儿也不为过。对 IT 运维工程师来说，构建一个真正可用的监控告警系统是一项艰巨的任务。在监控系统的开源软件中，可供选择的工具众多，然而真正符合自己需求，能够真正解决自己业务问题的监控系统软件却凤毛麟角。

笔者在运维从业生涯中用过的监控系统有 Cacti、Nagios，以及笔者公司开发的监控告警系统，直到接触了 Zabbix，才发现这个灵活而强大的自动化监控工具正是笔者以前所寻找的。Zabbix 灵活的设计为用户提供了易用的二次开发接口，让用户既可以使用 Zabbix 本身提供的功能，又可以自定义更多的监控项功能，从硬件监控，到操作系统，再到服务进程，以及网络设备，其无所不能的监控功能令人叹为观止，笔者不禁要为这么优秀的开源工具而震撼。

在 Zabbix 的使用过程中，也会遇到很多问题，包括：配置使用、架构设计、性能调优、大规模部署和应用等。由于 Zabbix 是一个集众多功能于一体的工具，自带的功能实在太多，一般用户往往仅用到其基本的功能，对更高级的功能并未使用到。然而随着使用的加深，会遇到更多的问题，这其中最大的问题是告警和数据存储。本书对这两部分内容都有深入的讲解。

笔者从 2012 年 12 月开始学习并使用 Zabbix，在此过程中，通过对 Zabbix 官网的学习，将 Zabbix 的部署和配置，以及其丰富的功能整理成文档，并对外公开，已在网络上公开过两个版本《Zabbix 使用手册 V1.4》和《Zabbix 使用手册 V2.0》，对不少学习 Zabbix 的朋友有所帮助。为了将 Zabbix 的功能更加详细地展示出来，于是有了本书的出现。本书以简洁通俗的方式将 Zabbix 的各项功能展现给读者，使读者即学即用，以此来节省读者宝贵的时间。

本书将不会开源电子书 PDF 版本，但笔者将来会对《Zabbix 使用手册》做更多的更新，本书的所有代码和软件是开源的。另外，笔者开源的 Zabbix 安装包定制项目对重新打包 Zabbix 的 RPM 安装包有一定的参考价值，其地址为 <https://github.com/itnihao/zabbix-rpm>，希望对大家有所帮助。

如何阅读本书

本书从运维（OPS）角度对 Zabbix 的各项功能进行了详细介绍，以自动化运维视角为出发点，对自动化功能、高可用、监报告警、性能调优、Zabbix API、Zabbix 协议、RPM 安装包定制等进行了深入浅出的探讨。

第 1 部分为基础部分，包括第 1~6 章，介绍监控系统的原理，让初次接触监控的读者了解监控的组成架构，从宏观上认识监控系统，接下来讲解了 Zabbix 的架构、Zabbix 的安装、监控配置、自定义监控项、告警配置、告警脚本等功能。这部分内容适合初学者系统地学习 Zabbix 监控系统，对稍有经验的使用者来说，重点掌握 3.9 节中 Zabbix 对数据存储的处理，以及第 5 章和第 6 章的内容。

第 2 部分为中级部分，包括第 7~11 章，对 Zabbix 的多种监控方式进行了详述（如 IPMI、SNMP、JMX 等），对 Zabbix 的自动化功能进行了深入讲解，介绍了分布式监控系统，并有大量自定义脚本的监控案例，对自动化发现 LLD 功能进行了深入的讲解，适合对 Zabbix 有一定基础的读者深入学习。通过对这部分的学习，读者将会深入理解 Zabbix 的自动化功能。

第 3 部分为高级部分，包括第 12~16 章，介绍 Zabbix 的性能调优，解决 Zabbix 在使用过程中遇到性能瓶颈的问题，并让读者学会使用 Zabbix API、Zabbix 协议来扩展 Zabbix，从而为构建运维平台提供了更多的扩展性。同时，也讲解了如何定制 RPM 包，以及如何使用 SaltStack 来自动化部署和配置，最后用一个实例来讲解如何构建企业级分布式监控系统，适合对 Zabbix 非常熟悉的读者进行深入应用。通过对这部分的学习，读者应该能够深入理解 Zabbix。

第 4 部分为附录部分，介绍 Zabbix 的源码安装和 Zabbix 的升级，仅供读者参考。在介绍 Zabbix 的安装时，作者极力推荐在 RHEL 这类系统中使用 RPM 包的安装方式，但网络中大部分资料都采用源码安装，且在多种书籍中均有使用源码安装的习惯，从而误导了读者，认为源码安装才是真正地懂 Linux，以此造成部分用户以使用源码安装为荣，让源码安装具有“优越感”。使用源码安装导致部分初学者连最基本的安装和配置都会遇到很多麻烦，甚至安装一个软件都需要几天才能完成。当然，这里并不排斥使用源码安装，笔者只是为了纠正一个观念，请读者尽量使用自己打包的 RPM 这种适合大规模安装的方式去部署 Zabbix 程序。本书将 RPM 定制的内容作为单独的一章进行介绍，让读者深入理解安装和定制的过程。

读者对象

- 中/高级 Linux 系统管理员
- 系统运维工程师
- 运维开发工程师

- 系统集成商
- 运维监控系统工程师
- 监控系统软件开发（设计）人员
- IT 管理人员
- 架构设计人员

勘误和支持

因作者水平有限，书中的错误或不妥之处难免，恳请读者批评、指正。如果读者有任何宝贵的意见或建议，可以发送邮件到 zabbix@itnihao.com，我们将尽快给予反馈。

本书所有的代码和安装软件将放在 GitHub 中托管，地址为 <https://github.com/itnihao/zabbix-book/>，读者可以自行下载并使用。另外，本书的勘误也会在该链接中得到反馈。

声明

本书采用的 Zabbix 版本为 2.0 和 2.2，因此对 Zabbix 1.8 不再讨论，Zabbix 2.4 将在未来两年内会发布，其功能与 Zabbix 2.2 将有所不同（例如，Zabbix 2.4 中的分布式模式会去掉 node 架构等）。本书采用的操作系统以 RHEL（CentOS）、Windows 为主，对于其他系统，其配置方法类似，请读者举一反三。本书对部分不重要的内容只是简略地介绍，希望能起到抛砖引玉的作用。

另外，本书在编写过程中，参考了以下网址中的内容，并对官方网站的部分内容进行了翻译和整理，后面不再单独声明。

```
https://www.zabbix.com/documentation/2.2/manual
https://www.zabbix.org/wiki/
http://pengyao.org/zabbix-triggers-functions.html
```

鸣谢

本书在编写过程中得到了众多朋友的支持。在这里要感谢姚炫伟、陈艺超、窦喆、冯颖聪等对本书的技术问题进行审校；感谢电子工业出版社编辑任晓露和李利健对本书的精心指导，更正了书稿中的很多错误，本书才得以与读者见面。特别感谢以下朋友对作者的支持与鼓励：伊杨林、吴华、黄小路、芮峰云、罗坤、温宏强、曾兵、范仁更、邝玲、张克元（网名 Geek）、沈灿、薛群、李佃田、罗苗、杨晨、焦婷婷、KissPuppet（网名）、司鼎任、汤永全、罗伟、水喜云、唐文军、李彬（网名彬彬）、高鹏程、唐博、罗涛（网名 MorningSong）等。注意，以上排名不分先后（使用 Linux 命令 `sort -R` 随机生成排序）。感谢 SJCloud 公司的全体成员，感谢公司领导金剑、陶永国、付自强在笔者工作期间给予的支持和帮助。

另外，也要感谢史应生、李庆雷、姚仁捷、邓磊、马哥（马永亮）、孙科伟、虚拟的现实（网名）等分享的有关 Zabbix 的技术文档、视频录制和讲座等内容，这些内容对促进 Zabbix 在国内的推广和发展有很大帮助。很多网友也写了大量有关 Zabbix 的博客文章，相关链接会放在本书的 GitHub 项目中。

本书的代码示例规范

(1) 在 Shell 环境中使用，Shell 命令用黑体加粗。

```
shell# vim /etc/php.ini
```

(2) 在 MySQL 环境中使用，SQL 命令用黑体加粗。

```
mysql> flush privileges;
```

(3) 在本书中，Zabbix-Server 表示 zabbix-server 服务或进程，其他 Zabbix-Agent、Zabbix-Get、Zabbix-Proxy 情况类似；代表 zabbix-server、zabbix-agent、zabbix-get、zabbix-proxy、zabbix-sender 等程序名称本身时，一律采用小写；程序配置文件中的字段，如 Hostname=Zabbix proxy、Hostname=Zabbix server，统一采用原配置文件中的风格，后面不再声明。

(4) 重要参数的解释，用灰色底纹表示。

- Proxy name, 即Zabbix-Proxy的hostname。
- Proxy mode, 即Zabbix-Proxy的工作模式，被动或主动，默认是主动模式。

作者（网名 itnihao）

目 录

第 1 部分 基础部分

第 1 章 监控系统简介	2
1.1 为何需要监控系统	2
1.2 监控系统的实现	2
1.3 监控系统的开源软件现状	4
1.3.1 MRTG	4
1.3.2 Cacti	5
1.3.3 SmokePing	5
1.3.4 Graphite	6
1.3.5 Nagios	7
1.3.6 Zenoss Core	7
1.3.7 Ganglia	8
1.3.8 OpenTSDB	9
1.3.9 Zabbix	10
1.4 监控系统的原理探究	11
第 2 章 Zabbix 简介	14
2.1 Zabbix 的客户	14
2.2 使用 Zabbix 的准备	15
2.3 Zabbix 为何物	15
2.4 选择 Zabbix 的理由	16
2.5 Zabbix 的架构	17
2.6 Zabbix 的运行流程	18
2.7 Zabbix 的功能特性	19
第 3 章 安装与部署	21
3.1 安装环境概述	21

3.1.1	硬件条件	21
3.1.2	软件条件	23
3.1.3	部署环境的考虑	25
3.2	Zabbix-Server 服务器端的安装	25
3.2.1	安装 Zabbix-Server	26
3.2.2	安装 MySQL 数据库服务	27
3.2.3	配置 zabbix_server.conf	28
3.2.4	防火墙、Selinux 和权限的设置	30
3.2.5	配置 Web 界面	32
3.2.6	故障处理	37
3.3	Zabbix-Agent 客户端的安装	39
3.3.1	安装 Zabbix-Agent	39
3.3.2	防火墙的设置	39
3.3.3	配置 zabbix_agentd.conf	39
3.4	SNMP 监控方式的配置	40
3.5	在 Windows 中安装 Zabbix-Agent	40
3.6	其他平台的安装	43
3.7	Zabbix-Get 的使用	43
3.8	Zabbix 相关术语（命令）	44
3.9	Zabbix-Server 对数据的存储	45
3.9.1	Zabbix 对数据存储	45
3.9.2	MySQL 表分区实例	51
3.10	Zabbix init 脚本解释	55
3.11	高可用和安全	56
3.12	Zabbix 数据库的备份	57
第 4 章	快速配置和使用	59
4.1	配置流程	59
4.2	主机组的添加	61
4.3	模板的添加	63
4.4	添加主机	65
4.5	Graphs 的配置	68
4.6	Screen 的配置	74
4.7	Slide shows 的配置	78
4.8	Zatree 的使用	79

4.9 Map 的配置	80
4.10 Web 监控	85
4.10.1 Web 监控的原理	85
4.10.2 Web 监控指标	85
4.10.3 Zabbix 中 Web 监控的配置	86
4.10.4 认证的支持	89
4.10.5 触发器的设置	91
4.10.6 排错	91
4.11 IT 服务	92
4.12 报表	95
4.13 资产管理	97
第 5 章 深入配置和使用	99
5.1 Items 的添加	99
5.1.1 Items 的含义	99
5.1.2 如何添加 Items	99
5.2 Items key 的添加	105
5.3 Items 的类型	109
5.3.1 Zabbix-Agent	109
5.3.2 Simple check	113
5.3.3 日志监控方式	115
5.3.4 监控项计算 (Calculated)	120
5.3.5 聚合检测 (Aggregate)	124
5.3.6 内部检测 (Internal)	127
5.3.7 SSH、Telnet 和扩展检测	128
5.4 宏的配置	129
5.5 维护时间	131
5.6 事件确认	132
5.7 数据的导入/导出配置	134
第 6 章 告警和配置	135
6.1 告警概述	135
6.2 Trigger 的配置	136
6.2.1 Trigger 的状态	136
6.2.2 Trigger 的配置步骤	136

6.2.3	Trigger 告警依赖	141
6.2.4	Trigger 正则中的单位	141
6.2.5	Trigger 表达式举例	142
6.2.6	Trigger 函数	146
6.3	添加 Actions	151
6.3.1	Actions 概述	151
6.3.2	Actions 的配置	152
6.3.3	Conditions 的配置	155
6.3.4	Operations 的功能	156
6.3.5	告警消息发送的配置	156
6.3.6	执行远程命令的配置	158
6.4	邮件告警配置的实例	160
6.4.1	创建 Media	160
6.4.2	创建用户	161
6.4.3	创建 Actions	162
6.5	自定义脚本告警	163
6.6	邮件告警脚本的配置实例	165
6.7	告警升级的机制	169
6.8	告警配置故障排查	172

第2部分 中级部分

第7章	监控方式剖析	176
7.1	Zabbix 支持的监控方式	176
7.2	Zabbix 监控方式的逻辑	177
7.3	Agent 监控方式	177
7.4	Trapper 监控方式	177
7.4.1	Trapper 的配置步骤	178
7.4.2	Trapper 的配置示例	178
7.4.3	使用 zabbix_sender 发送数据	179
7.5	SNMP 监控方式	180
7.5.1	SNMP 概述	180
7.5.2	SNMP 协议的运行	181
7.5.3	SNMP 协议原理	181
7.5.4	MIB 简介	184

7.5.5	SNMP 的相关术语	186
7.5.6	配置 Zabbix 以 SNMP 方式监控	186
7.6	IPMI 监控方式	189
7.7	JMX 监控方式	194
7.7.1	JMX 在 Zabbix 中的运行流程	195
7.7.2	配置 JMX 监控的步骤	195
7.7.3	安装 Zabbix-Java-Gateway	195
7.7.4	配置 Zabbix-Java-Gateway	196
7.7.5	监控 Java 应用程序	196
7.7.6	自定义 JMX 的 Key	197
7.7.7	监控 Tomcat	199
7.7.8	Weblogic 的监控	200
7.8	命令的执行	201
第 8 章	分布式监控	202
8.1	代理架构	202
8.2	节点架构	205
8.3	被动模式和主动模式	206
8.3.1	被动模式	206
8.3.2	主动模式	207
第 9 章	Zabbix 与自动化运维	211
9.1	监控自动化	211
9.2	网络发现	212
9.3	主动方式的自动注册	215
9.3.1	功能概述	215
9.3.2	主动方式自动注册的配置	215
9.3.3	使用 Host metadata	219
9.3.4	关于自动注册的注意事项	221
9.4	Low level discovery 功能	222
9.4.1	现实案例需求	224
9.4.2	Zabbix 客户端配置	225
9.4.3	Low level discovery 自动发现脚本编写	225
9.4.4	自定义 Key 配置文件	227
9.4.5	Web 页面添加 Low level discovery	228

9.5 Zabbix 与自动化配置管理工具 SaltStack	238
第 10 章 使用的经验和技巧	242
10.1 如何有效地设置监控告警	242
10.2 监控项的使用技巧	246
10.3 触发器的使用技巧	246
10.4 触发器配置	247
10.5 谷歌浏览器告警插件	249
10.6 数据图断图	250
第 11 章 监控案例	252
11.1 监控 TCP 连接数	252
11.2 监控 Nginx	254
11.3 监控 PHP-FPM	256
11.4 监控 MySQL	260
11.4.1 用自带的模板监控 MySQL	260
11.4.2 用 Percona Monitoring Plugins 监控 MySQL	265
11.6 监控 DELL 服务器	272
11.7 监控 Cisco 路由器	272
11.8 监控 VMware	275

第 3 部分 高级部分

第 12 章 性能优化	282
12.1 Zabbix 性能优化概述	282
12.2 Zabbix 性能优化的依据	283
12.3 配置文件的参数优化	285
12.4 Zabbix 的架构优化	287
12.5 Items 工作模式及 Trigger 的优化	287
12.6 Zabbix 的数据库优化	287
12.7 其他方面	289
第 13 章 Zabbix API 的使用	290
13.1 Zabbix API 简介	290
13.2 JSON-RPC	290

13.3 Zabbix API 的使用流程	293
13.3.1 使用 API 的基本步骤	293
13.3.2 如何使用官方文档获取帮助	293
13.3.3 用 CURL 模拟 API 的使用	294
13.3.4 HTTP 头部 Content-Type 设置	296
13.3.5 关于用户认证	296
13.3.6 获取主机信息（用 Python 写的示例）	297
13.3.7 添加 Host	299
13.3.8 删除 Host	301
13.4 第三方 Zabbix API 模块	302
第 14 章 使用 Zabbix 协议	304
14.1 Zabbix 协议概述	304
14.2 Zabbix-Sender 协议	305
14.2.1 Sender 数据发送	306
14.2.2 Server 对数据响应的处理	307
14.2.3 Zabbix-Sender 的实例	307
14.3 Zabbix-Get 协议	310
14.4 Zabbix-Agent 协议	310
第 15 章 定制 Zabbix 安装包	313
15.1 为什么要定制安装包	313
15.2 如何定制安装包	313
第 16 章 大型分布式监控案例	316
16.1 监控系统构建概述	316
16.2 监控环境架构图	317
16.3 架构实现的过程	317
16.3.1 硬件和软件需求	317
16.3.2 Zabbix DB 的安装	319
16.3.3 安装 Zabbix-Server	325
16.3.4 安装 Zabbix-GUI	327
16.3.5 安装 Zabbix-Proxy	332
16.3.6 配置 Zabbix-Agent	335
16.4 业务相关的配置	335

16.4.1 用户的配置	335
16.4.2 业务组的配置	336
16.4.3 监控模板的定制	336
16.4.4 自动发现的配置	338
16.5 其他需求	338
附录 A 源码安装及相关配置	339
A.1 安装 Zabbix-Server	339
A.2 安装 Zabbix-Agent	341
A.3 关于 Zabbix 的升级	342

全國統編教材 第一章

第 1 部分

基 础 部 分

第一章 緒論 1.1

第一章「緒論」介紹本書之編纂經過，說明本書之編纂目的及範圍，並介紹本書之編纂體例。第二章「基礎部分」介紹本書之編纂體例，說明本書之編纂目的及範圍，並介紹本書之編纂體例。

第1章 监控系统简介

本章阐述了监控系统的发展历程、监控系统的原理，以及监控系统的实现过程，对现有的开源监控解决方案进行了综合分析，目的是让读者全面了解监控系统，让读者更加深入地学习到监控系统的原理，为后面章节的学习做好准备。

1.1 为何需要监控系统

在一个 IT 环境中会存在各种各样的设备，例如，硬件设备、软件设备，其系统的构成也是非常复杂的，通常由如图 1-1 所示的模型构成。

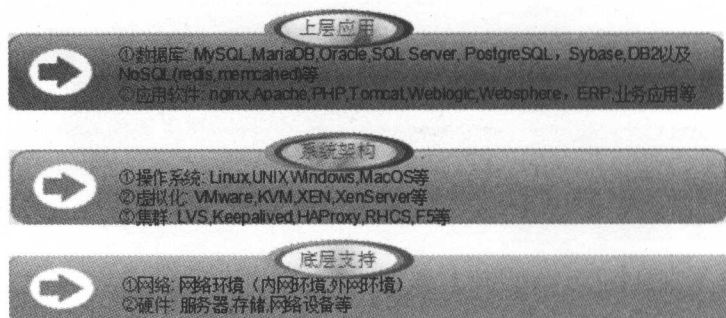


图 1-1

多种应用构成复杂的 IT 业务系统，保证这些资源的正常运转，是一个公司 IT 部门的职责。而要让这些应用能够稳定地运行，则需要专业 IT 人员进行设计、架构、维护和调优。在这个过程中，为了及时掌控基础环境和业务应用系统的可用性，需要获取各个组件的运行状态，如 CPU 的利用率、系统的负载、服务的运行、端口的连通、带宽流量、网站访问状态码等信息。而这一切都离不开监控系统。

1.2 监控系统的实现

一个监控系统的组成大体可以分为两部分：**数据采集部分（客户端）**和**数据存储分析告警展示部分（服务器端）**，如图 1-2 所示。这两部分构成了监控系统的基本模型。

数据采集的工作模式可以分为**被动模式**（服务器端到客户端采集数据）和**主动模式**（客户端主动上报数据到服务器端）。通常，大多数监控系统应该能同时支持这两种模式。被动模式对服务器的开销较大，适合小规模的监控环境；主动模式对服务器的开销较小，适合大规模的监控环境。

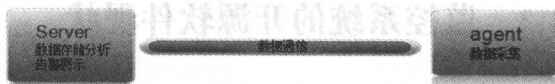


图 1-2

采集数据的协议方式可以分为两种：**专用客户端采集**和**公用协议采集**（SNMP、SSH、Telnet 等），如图 1-3 所示。

对于采集到的监控数据，可以将其存储到数据库或者文本或者其他方式，具体采用哪一种，应根据实际需求来决定。

怎么规划监控系统的架构设计呢？下面将详细分析。

对于一般的监控环境，被监控的节点不多，产生的数据较少，采用 C/S（Client/Server，客户端/服务器端）架构就足够了，如图 1-4 所示，这种架构适合于规模较小、处于同一地域的环境。

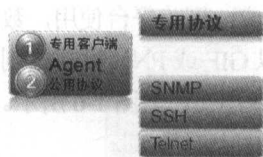


图 1-3

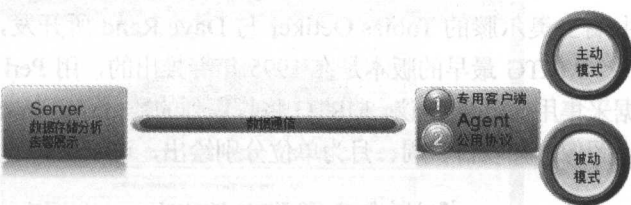


图 1-4

对于大规模的监控环境，被监控的节点多，且监控类型多，监控产生的数据和网络连接开销会非常巨大，而且由于跨地域等多种因素，需要分布式的解决方案，常见的方式为 C/P/S（Client/Proxy/Server，客户端/代理端/服务器端）架构（如图 1-5 所示），采用中间代理将大大提高监控服务器端的处理速度，从而能支撑构建大型分布式监控的环境。

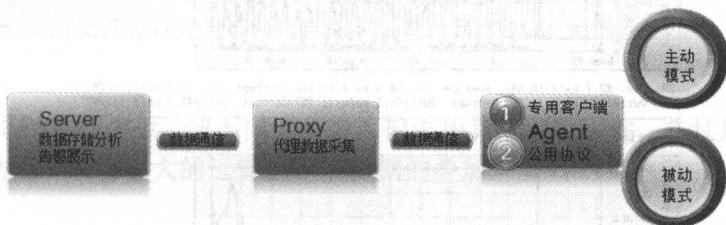


图 1-5

监控系统更重要的功能是告警和故障处理，这对及时解决问题和故障自愈非常重要。告警的时候，需要考虑到故障的有效汇报和集中汇报，防止出现“告警

洪水”，即同一类告警信息重复大量地发送。

1.3 监控系统的开源软件现状

前面简单介绍了监控的原理，下面看看已有的解决方案。

在监控软件中，开源的解决方案有流量监控（MRTG、Cacti、SmokePing、Graphite 等）和性能告警（Nagios、Zabbix、Zenoss Core、Ganglia、OpenTSDB 等）可供选择，并且每种软件都有自己的特点和功能，各自的侧重点和目标不完全相同，在设计理念和实现方法上也大同小异，但都具有共同特征，例如，采集数据、分析展示、告警以及简单的故障自动处理。最终都能达到对 IT 系统服务可用性的一个完全展示。下面将详细介绍各自的特点。

1.3.1 MRTG

MRTG（Multi Router Traffic Grapher）是一套可用来绘制网络流量图的软件，由瑞士奥尔滕的 Tobias Oetiker 与 Dave Rand 所开发，以 GPL 授权。

MRTG 最早的版本是在 1995 年春推出的，用 Perl 语言写成，可跨平台使用，数据采集用 SNMP 协议，MRTG 将收集到的数据通过 Web 页面以 GIF 或 PNG 格式绘制出图像，并以日、周、月为单位分别绘出，可以查询最大值和最小值，如图 1-6 所示。

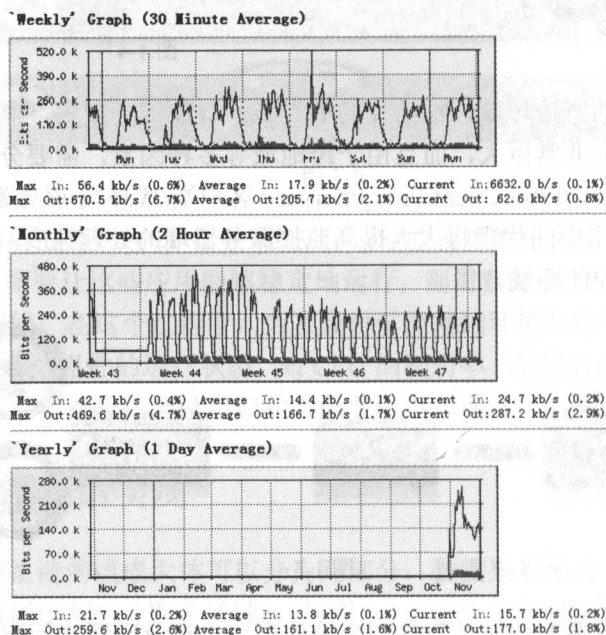


图 1-6

MRTG 原本只能绘出网络设备的流量图，后来发展出了各种插件。因此，网络以外的设备也可由 MRTG 监控，例如，服务器的硬盘使用量、CPU 的负载等。

1.3.2 Cacti

Cacti（英文含义为仙人掌）是一套基于 PHP、MySQL、SNMP 和 RRDtool 开发的网络流量监测图形分析工具，如图 1-7 所示。它通过 snmpget 来获取数据，使用 RRDtool 绘图，但使用者无须了解 RRDtool 复杂的参数。它提供了非常强大的数据和用户管理功能，可以指定每一个用户能查看树状结构、主机设备以及任何一张图，还可以与 LDAP 结合进行用户认证，同时也能自定义模板，在历史数据的展示监控方面，其功能相当不错。

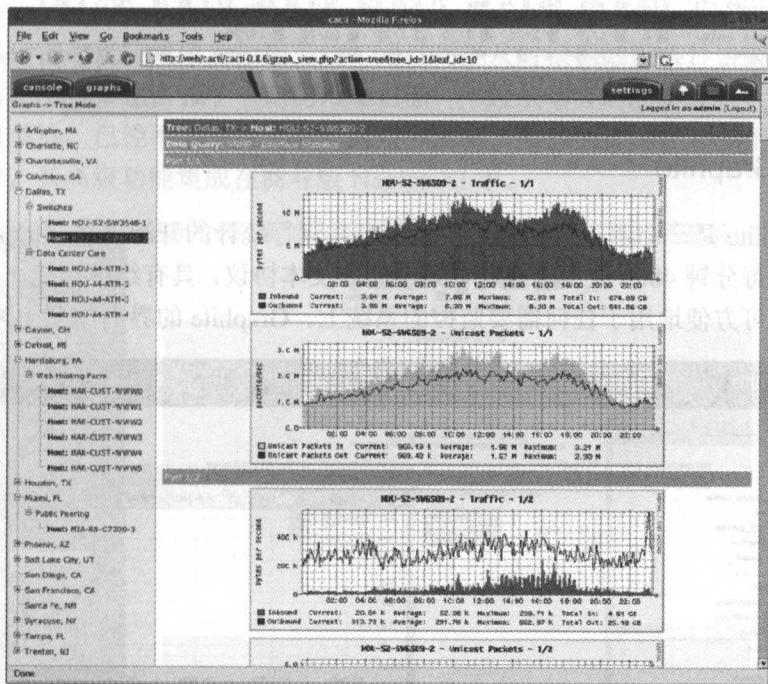


图 1-7

Cacti 通过添加模板，使不同设备的监控添加具有可复用性，并且具备可自定义绘图的功能，具有强大的运算能力（数据的叠加功能）。

1.3.3 SmokePing

SmokePing 主要用于监视网络性能，包括常规的 ping、WWW 服务器性能、DNS 查询性能、SSH 性能等。底层也是用 RRDtool 做支持，特点是绘制的图非常

漂亮，网络丢包和延迟用颜色和阴影来表示，支持将多张图叠放在一起，如图 1-8 所示。其作者（Tobi Oetiker）还开发了 MRTG 和 RRDtool 等工具。

SmokePing 的站点为：<http://tobi.oetiker.ch/hp/>。

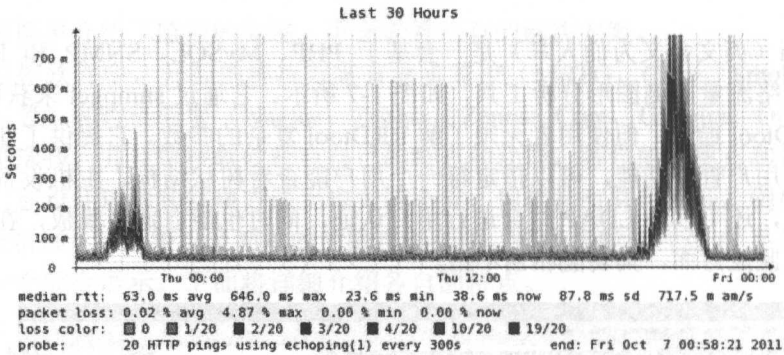


图 1-8

1.3.4 Graphite

Graphite 是一个用于采集网站实时信息并进行统计的开源项目。Graphite 服务支持平均每分钟 4800 次更新操作，采用简单文本协议，具有绘图功能，其即插即用的功能可方便地用于任何需要监控的系统上。Graphite 的界面如图 1-9 所示。

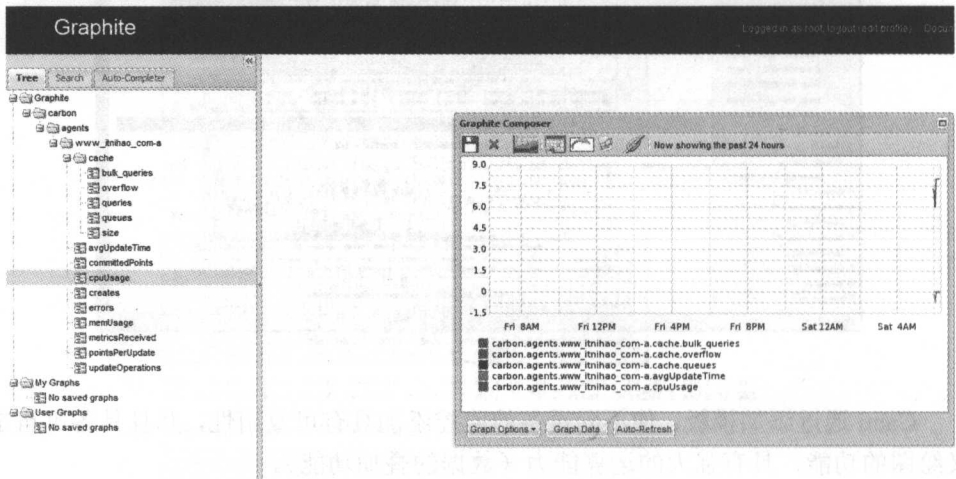


图 1-9

和其他监控工具不同的是，Graphite 本身并不收集具体的数据，这些数据收集的工作通常由第三方工具或插件完成（如 Ganglia、Nagios、collectd、statsd、Collectl 等）。因此，可以说，Graphite 是一个绘图工具。

值得一提的是，Graphite 用 Python 语言编写，采用 Django 框架，对于熟悉 Python 的用户（通常是运维人员）来说，将是一个不错的绘图工具选择。

简单地说，Graphite 做两件事：存储数据和按需绘图。

Graphite 的官方站点为：<http://graphite.wikidot.com/>。

1.3.5 Nagios

Nagios 是一个企业级的监控系统，可监控服务的运行状态和网络信息等，并能监视所指定的本地或远程主机参数以及服务，同时提供异常告警通知功能等。

Nagios 可运行在 Linux 和 UNIX 平台上，同时提供一个可选的基于浏览器的 Web 界面，以方便系统管理人员查看网络状态、各种系统问题，以及日志等，如图 1-10 所示。

Nagios 的功能侧重于监控服务的可用性，能及时根据触发条件告警。

目前，Nagios 也占领了一定的市场份额，不过从笔者的观察来看，Nagios 并没有与时俱进，已经不能满足于多变的监控需求，架构的扩展性和使用的便捷性有待增强，其高级功能集成在商业版 Nagios XI 中。

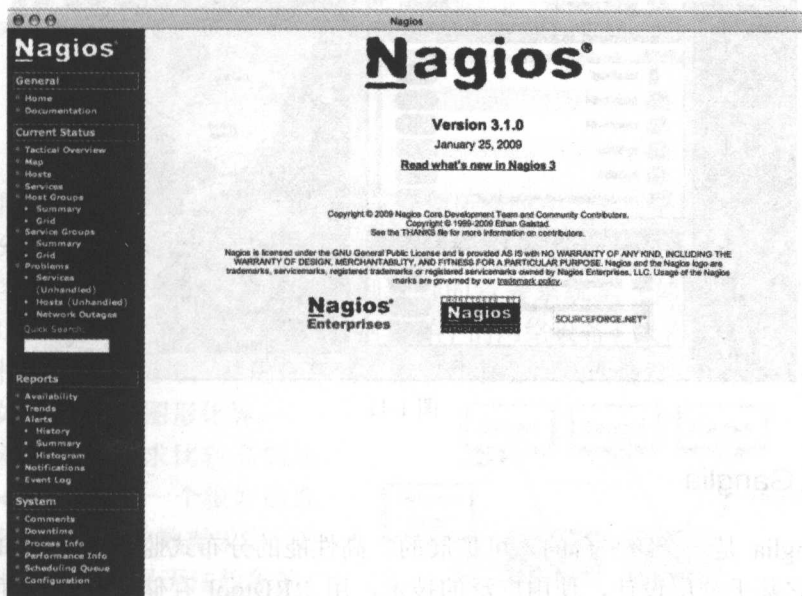


图 1-10

1.3.6 Zenoss Core

Zenoss Core（简称 Zenoss）是开源企业级 IT 管理软件，它允许 IT 管理员依

靠单一的 Web 控制台来监控网络架构的状态和健康度。

Zenoss Core 的强大功能来自深入的列表与配置管理数据库，用于发现和管理公司 IT 环境的各类资产（包括服务器、网络和其他结构设备）。Zenoss 可以创建关键资产清单和对应的组件级别（接口、服务、进程、已安装的软件等）。建立好模型后，Zenoss 就可以监控和报告 IT 架构中各种资源的状态和性能状况了。同时还提供与 CMDB 关联的事件和错误管理系统，以协助提高各类事件和提醒的管理效率，以此提高 IT 管理人员的效率。

Zenoss Core 采用 SNMP 来采集数据，其界面如图 1-11 所示。

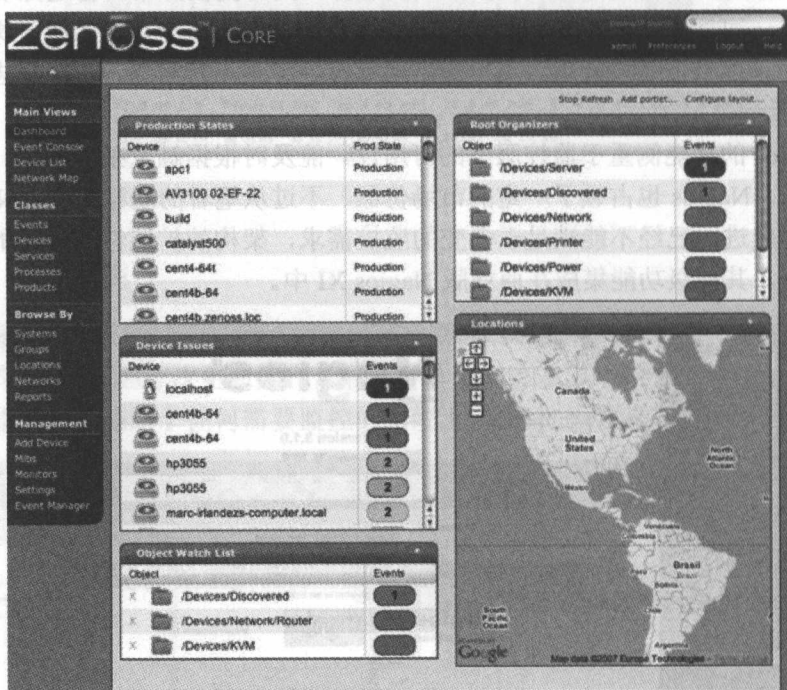


图 1-11

1.3.7 Ganglia

Ganglia 是一个跨平台的、可扩展的、高性能的分布式监控系统，如集群和网络。它基于分层设计，使用广泛的技术，用 RRDtool 存储数据，具有可视化界面，适合于对集群系统的自动化监控。其精心设计的数据结构和算法使得监控端到被监控端的连接开销非常低。目前已经有成千上万的集群正在使用这个监控系统，可以轻松地处理 2000 个节点的集群环境。该软件的部分截图如图 1-12 所示。

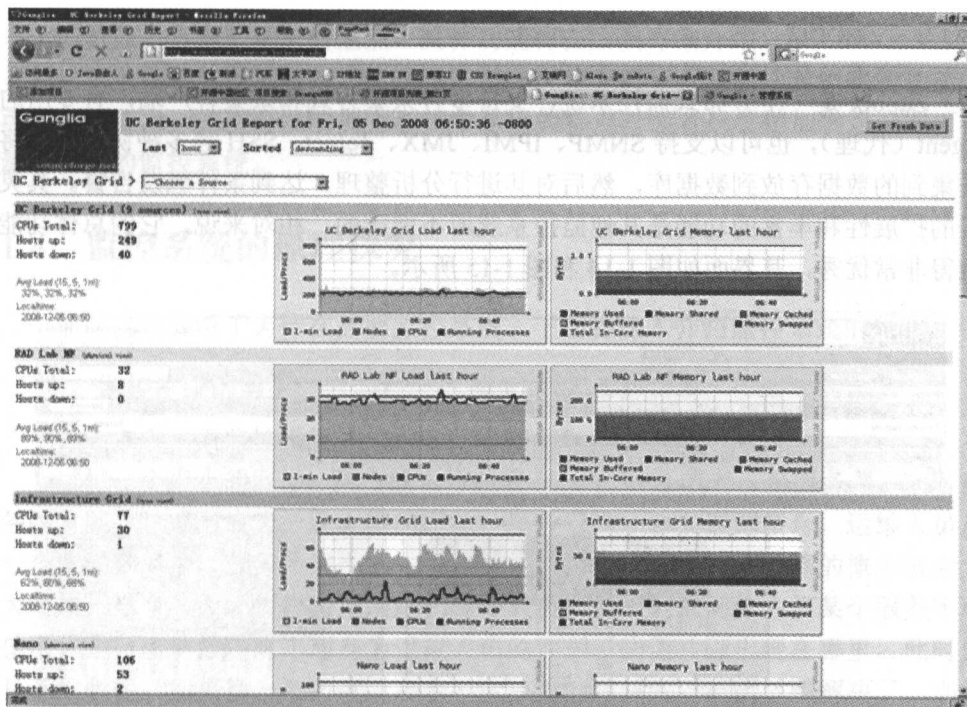


图 1-12

1.3.8 OpenTSDB

开源监控系统 OpenTSDB 用 HBase 存储所有时序（无须采样）的数据，来构建一个分布式、可伸缩的时间序列数据库。它支持秒级数据采集，支持永久存储，可以做容量规划，并很容易地接入到现有的告警系统里（如图 1-13 所示）。OpenTSDB 可以从大规模的集群（包括集群中的网络设备、操作系统、应用程序）中获取相应的采集指标，并进行存储、索引和服务，从而使这些数据更容易让人理解，如 Web 化、图形化等。

在对实时性要求比较高的场合，OpenTSDB 是一个很好的选择。它支持秒级别的数据采集，这在其他监控系统中是无法想象的。因得益于其存储系统的选择，所以它支持大数据分析。因此，这个开源软件在未来的环境中会有更多的用户，也会获得更广泛的支持。

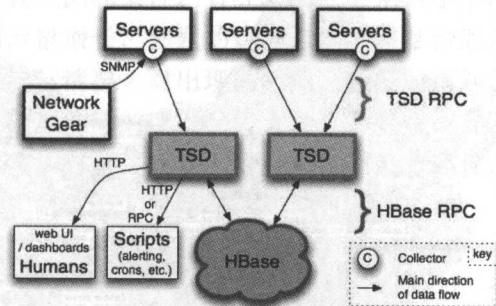


图 1-13

1.3.9 Zabbix

Zabbix 是一个分布式监控系统，支持多种采集方式和采集客户端，有专用的 Agent（代理），也可以支持 SNMP、IPMI、JMX、Telnet、SSH 等多种协议，它将采集到的数据存放到数据库，然后对其进行分析整理，达到条件触发告警。其灵活的扩展性和丰富的功能和其他监控系统所不能比的。相对来说，它的总体功能做得非常优秀，其界面如图 1-14 和图 1-15 所示。

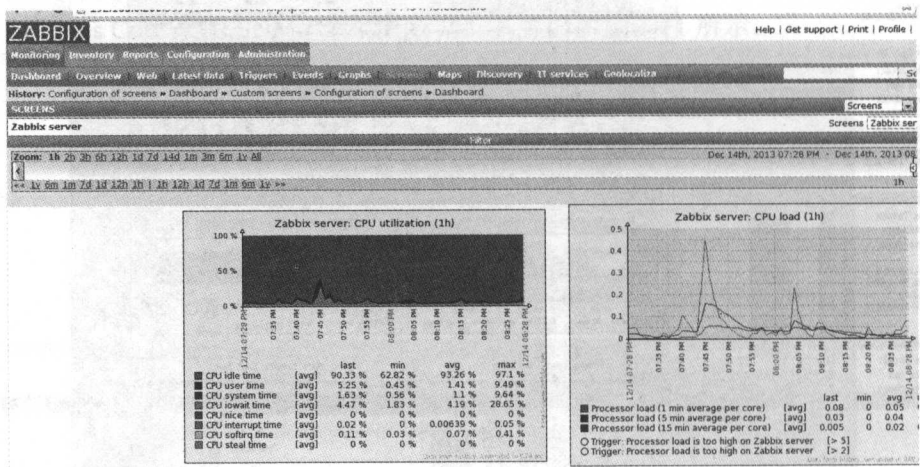


图 1-14

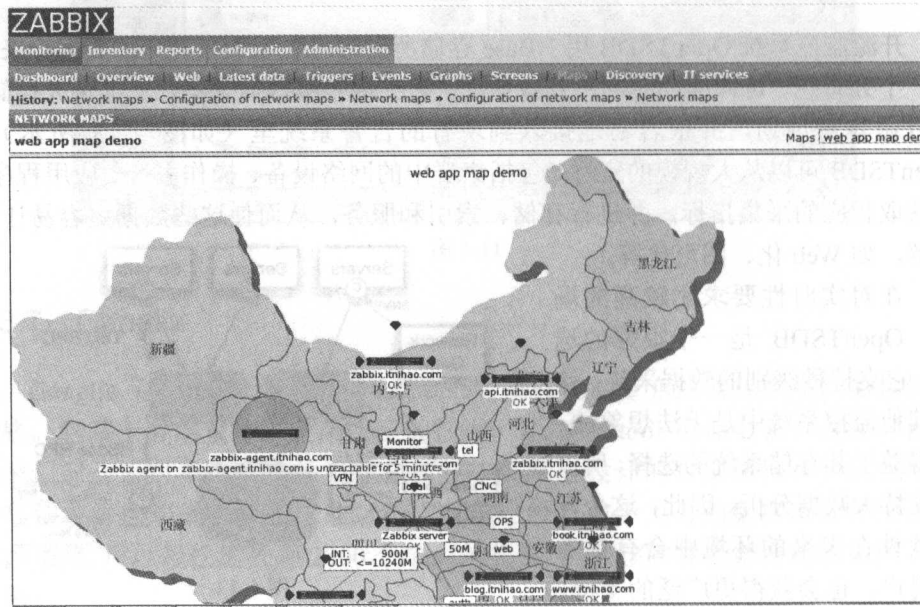


图 1-15

该监控工具也是本书的重点，具体功能将在后面详细介绍。

从以上各种监控系统的对比来看，Zabbix 都是具有优势的，其丰富的功能、可扩展的能力、二次开发的能力和简单易用的特点，读者只要稍加学习，即可构建起自己的监控系统。

1.4 监控系统的原理探究

前面已经介绍了为什么需要监控系统，下面将深入介绍监控系统的原理。

1. 监控系统的诞生

首先来看一下监控系统的使用者都有哪些？

试想一个很小的网站，刚上线也许就只有一个 VPS 或者一台服务器。随着业务的发展，它逐渐成为一个大型的网站，服务器从一台发展到多台，运维人员从一个发展到多个，业务战线也从一条发展到多条，这时候出现故障的概率就会大大增加。当有一天，你作为该网站的运维人员，你的老板问你为何某个服务不可使用，为何出现故障时，而身为运维人员的你却不知道故障出现在哪里，此时你是不称职的。如果换一个角度，在未发生故障的时候就已经把故障解决了，那你就是幕后英雄。

有一个故事是讲神医扁鹊的。有人对扁鹊说：“你是天下的神医，能治天下各种疑难杂症，很了不起。”而扁鹊却说：“你知道吗？我和我长兄比起来，可差远啦！我长兄给人治病，是治人于病未发之前，在稍见端倪的时候就能防患于未然，而我只能治人病于膏肓之中，往往是病人受尽折磨才来找我。”（引自魏文王问扁鹊——《鹖冠子》世贤第十六）

监控系统就是这么一个“神医”，救人于膏肓之中（发送故障告警，或者自动修复故障），在发生故障之前，监控系统中会隐约显现故障的前期迹象，这也适用于其他事物，即任何事物的发生必有其原因和条件。有经验的人会从小小的迹象中发现更大的问题，例如，突发的流量增长，突发的访问量增大，某台服务器的瞬间负载变高，都表明了即将出现异常情况。对出现的故障，能及时通知告警和故障的自动修复，对运维人员响应处理故障的速度会大大加快，甚至在异常严重的故障情况下，对及时采用应急预案，有不可或缺的作用。如果没有监控，故障的反馈往往来自用户的报告，然后才由运维人员处理。

2. 监控系统的实现

监控系统往往需要对物理硬件和应用软件的性能和参数进行数据汇集，实现集中管理和统一分析。

在一个监控系统中，构成要素为监控服务器端程序、数据存储、被采集节点

等相关模块，其告警分析和自动故障处理功能由服务器端执行。在数据采集完成之后，需要对采集到的数据进行分析 and 处理，判断是否有异常，是否属于告警条件。告警条件如何设置呢？通常是根据实际的经验值、业务需求来设置告警阈值。达到告警条件时，则发送告警信息给管理人员，然而，对于有些故障，我们希望程序能自动处理，减少人工干预，让程序自动修复，只在出现严重故障、程序无法判断的时候，才告警通知管理人员处理。

一个监控系统往往需要集成资产管理，可以从逻辑上展示业务和功能的信息，通过对其进行数据分析，做到对投资与回报的一个反馈展示，为资产的合理规划与使用提供了依据。

从其工作模式来看，监控系统的数据采集可以分为两种：主动监控和被动监控，在 1.2 节已经详述，一个理想的监控系统，其采集端支持的采集方式越多，其扩张能力越强大，适用的环境场合越多。

监控系统需要提供一个 API，方便其他功能系统对监控数据进行操作管理，这在业务系统精密的情况下显得特别重要，通常能对外提供 API 功能的软件，其用户群会更广，产品会越做越好。API 一般可以分为 RESTful、SOAP 等形式，数据类型有 JSON、XML 等多种。从目前的趋势来看，RESTful 已经成为绝大多数 API 首选的方式。

监控系统需要对故障数据进行分析汇总，从故障中分析出现的概率，从而可以积累经验，避免以后出现类似的问题。例如，由于机器硬件导致的故障，其概率有多大，哪些部件最容易出问题，出问题的影响概率多大，问题解决的概率有多大。从监控的数据中就可以分析并发现相关数据，在此基础上进行分析汇总，可以整理出相应的对策和相应的技术应急方案。

常见的监控系统性能采集指标如表 1-1 所示。

表 1-1

监 控 项 目	详 细 内 容
主机监控	CPU、内存、磁盘的剩余空间/利用率和 I/O、SWAP 使用率、系统 UP 时间、进程数、负载
网卡监控	Ping 的往返时间及包成功率、网卡流量，包括流入/流出量和错误的数据包数
文件监控	监控文件大小、Hash 值，匹配查询、字符串存在与否
URL 监控	监测指定 URL 访问过程中的返回码、下载时间及文件大小，支持内容匹配
应用程序	端口和内存使用率、CPU 使用率、服务状态、请求数、并发连接数、消息队列的字节数、Client 事务处理数、Service 状态等
数据库	监测数据库中指定的表空间、数据库的游标数、Session 数、事务数、死锁数、缓冲池命中率、库 Cache 命中率、当前连接数、进程的内存利用率等性能参数
日志	错误日志匹配、特定字符串匹配
硬件	温度、风扇转速、电压等

3. 监控系统对时间的要求

监控系统需要根据实际应用的需求，实时/非实时地采集和展示数据。另外，包括历史趋势数据的展示和分析，以及容量报表、可用性报告的生成。

4. 监控系统的告警需求

支持多种方式，如短信、邮件、IM 和其他接口。具备可定制化功能，对第三方告警介质提供可编程接口。这一点在很多场合非常重要，例如，将告警结果发送到专用的告警分析系统。

支持对告警内容的分析自动处理，防止误报、漏报，以及防止抖动。这一点对大多数监控系统都是一个值得挑战和研究的课题。例如，一个机房网络发生故障，按照常规告警内容，会收到无数条告警信息，内容是每个设备的故障，而对于更高级的告警信息，我们希望收到的是“某机房存在网络故障，受影响的设备的 IP 是 X.X.X.X~X.X.X.X，受影响的业务是 XXX.”，这样做的目的是让告警信息更智能、更有效，防止“告警炸弹”的产生。

简而言之，监控数据的采集、存储、分析和故障报告是每个监控系统的基本功能，其他复杂的附加功能则是监控系统的增值业务。

在本书中，你将会看到以上功能的具体体现，无论你是使用 Zabbix，还是自行开发监控系统，本书都具有良好的参考价值。

第 2 章 Zabbix 简介

本章对 Zabbix 的发展历史、客户群、开源模式、软件架构和 Zabbix 的详细功能等方面进行了详细介绍,使读者对 Zabbix 有一个全面的认识,读者稍做了解即可,重点为如图 2-5 所示的 Zabbix 架构理解。

2.1 Zabbix 的客户

在国外,有如图 2-1 所示的用户在使用 Zabbix。

在国内,BAT 的部分业务、豆瓣、58 同城、PPTV、搜狐、Letv、人人、网易、小米、360 等公司都在使用 Zabbix。



图 2-1

注:对于以上信息,读者可以参考 <http://www.zabbix.com/users.php>。

随着云计算、虚拟化的大规模应用,以及未来移动互联网、物联网等的兴起,Zabbix 的使用将越来越广泛,应用场合也越来越多。目前,不少互联网公司、云计算公司、系统集成软件公司、外包服务公司等,都有对 Zabbix 进行二次开发和大规模使用。所以,可以断言,Zabbix 在未来将会引领监控软件的潮流。

Zabbix 适合中小型企业、大中型企业的用户使用。单个 Server 节点可以支持

上万台设备，每秒可以处理 1.5 万次请求，理论上可以支持 5 万台设备。

Zabbix 自身的定位是中型企业和大型企业，如果在特大型环境中（如图 2-2 所示的实例，是具有 10024 台设备规模的一个 Zabbix 环境）使用，需要解决大并发、大压力的问题，这对使用者提出了更高的要求。

STATUS OF ZABBIX		
Parameter	Value	Details
Zabbix server is running	Yes	localhost:10051
Number of hosts (monitored/not monitored/templates)	10024	10001 / 0 / 23
Number of items (monitored/disabled/not supported)	450081	450077 / 0 / 4
Number of triggers (enabled/disabled)[problem/unknown/ok]	170041	170041 / 0 [10057 / 0 / 159984]
Number of users (online)	2	1
Required server performance, new values per second	1439.4	-

图 2-2

2.2 使用 Zabbix 的准备

Zabbix 使用起来很简单，读者稍加学习即可上手，如果你以前从未接触过任何监控系统，从零基础开始学习，也是一件相对容易的事情。

这里暂且把 Zabbix 的用户分为如下三类（这样分不一定恰当）。

- 入门用户：以前从未接触过任何监控系统，也不懂Linux。学会Zabbix的安装和配置即可（安装可以用RPM，源码安装也很容易）。
- 中级用户：有Linux基础，熟悉LAMP和LNMP环境搭建、MySQL数据库、Shell脚本，以及简单的英文阅读能力，主要难点在于触发器、数据库调优和API的使用。
- 高级用户：熟悉PHP、C等一门脚本语言，具备二次开发能力，能修改源码，对Zabbix从代码级别进行优化和扩展。前提是对Zabbix的各个功能十分熟悉，或者至少熟悉需要改进的功能。

对一般用户和运维人员来说，达到中级用户使用水平，即可满足基本的使用需求。本书主要讲解前两类用户需要的知识，对第三类用户所需的知识也有涉及。

2.3 Zabbix 为何物

Zabbix 官方网站信息如下：

Zabbix官网	http://www.zabbix.com
官方论坛	http://www.zabbix.com/forum/
代码托管地址	http://sourceforge.net/projects/zabbix/files/
Wiki	https://www.zabbix.com/wiki
Zabbix体验地址	https://zabbix.org/zabbix/dashboard.php

Zabbix 是一个企业级的高度集成开源监控软件，提供分布式监控解决方案，可以用来监控设备、服务等可用性和性能，其产品不分企业版和社区版，是一个真正的源代码开放产品，用户可以自由下载并使用该软件。

Zabbix SIA 公司是 Zabbix 的官方技术团队成立的公司，其运作模式是商业软件的开源——软件的使用免费，服务收费。其为用户提供咨询、技术支持服务（定制开发、解决方案、人员培训等），如图 2-3 所示。



图 2-3

注：该图来自史应生的 Zabbix 讲座《Zabbix 监控概述》中的 PPT 文件截图。

2.4 选择 Zabbix 的理由

对比同类监控产品，有以下理由选择使用 Zabbix。

- ① Zabbix 是一个自由开放源代码的产品，用户可以对源代码进行任意修改和二次开发。Zabbix 采用 GNU General Public License (GPL) version 2 开源协议。
- ② 安装和配置简单，用户仅仅需要一些简单的学习，即可完成监控的搭建工作。
- ③ 搭建环境简单，基于开源软件构建平台，仅需要 Linux、Apache/Nginx、MySQL/PostgreSQL/Oracle、PHP 即可，无须专用操作系统支持，也无须专用硬件。
- ④ Zabbix-Agent 完全支持 Linux、UNIX、Windows、AIX、BSD 和 Solaris 的监控，Server 和 Agent 都采用 C 语言编码，对系统的资源占用非常小，数据采集的性能和速度非常快。

⑤ 将数据采集持久存储到数据库，便于对监控数据的二次分析。

⑥ 非常丰富的扩展能力，很轻松地自定义监控项和实现数据采集，几乎能监控所有的数据。例如，可以监控网站的访问次数，监控 UPS 和天气温度等。毫不夸张地说，在 Zabbix 的世界里，往往有你想不到的事情，没有办不到的事情。

⑦ 开源社区的运作模式，有各种论坛、邮件列表、IM 及时沟通等。

因此，如果你是一个系统管理员、网络管理员和运维人员，想要构建一套自己的监控系统环境，Zabbix 将会是最佳的选择。如果你是开发人员，想基于开源软件开发一套属于自己的监控系统，Zabbix 也是比较好的选择。

2.5 Zabbix 的架构

Zabbix 的版本大致可以分为：1.0、1.1、1.4、1.6、1.8、2.0、2.2 这几大发行版本（如图 2-4 所示）。对外的首个发行版本为 Zabbix 1.0 alpha1，其发行时间为 2001 年 4 月 7 日，随着版本的升级，其功能特性也在不断增强和完善。从 Zabbix 1.8 开始，增强了很多新的功能和特性，如对 JMX、自动发现、Low level discovery 功能的支持，而当前的 Zabbix 2.2 则有了进一步的完善，性能相比 Zabbix 2.0 提高了 5 倍，成为当前功能最强大的版本，支持对 VMware 的监控，可以动态加载模板，Web 能够模板化配置。下一个版本是 2.4，目前 Zabbix 官方正在规划中。

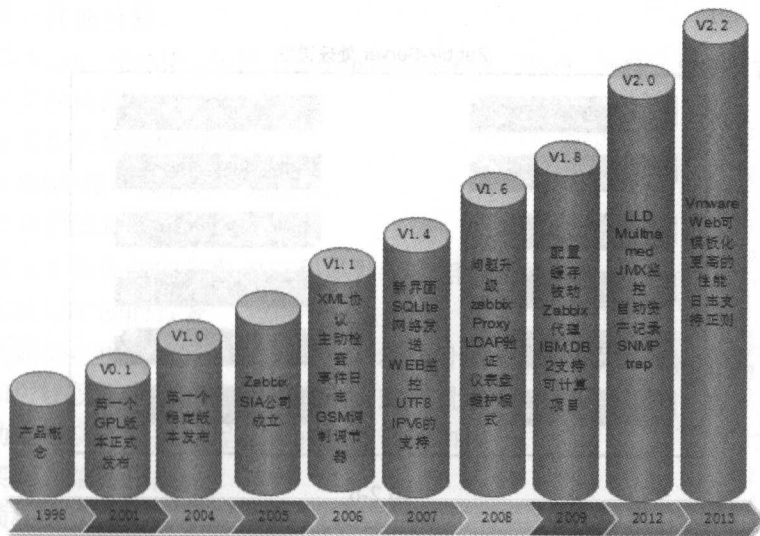


图 2-4

Zabbix 的通用架构是 Client/Server 架构，分布式架构为 Client/Proxy/Server 或 Client/Node/Server，Zabbix-Server 将采集到的数据持久地存储到数据库中，用

前端 UI 友好地展示给用户，图 2-5 展示的是用 Agent 采集数据，Proxy 做代理，Server 处理数据的架构模式。

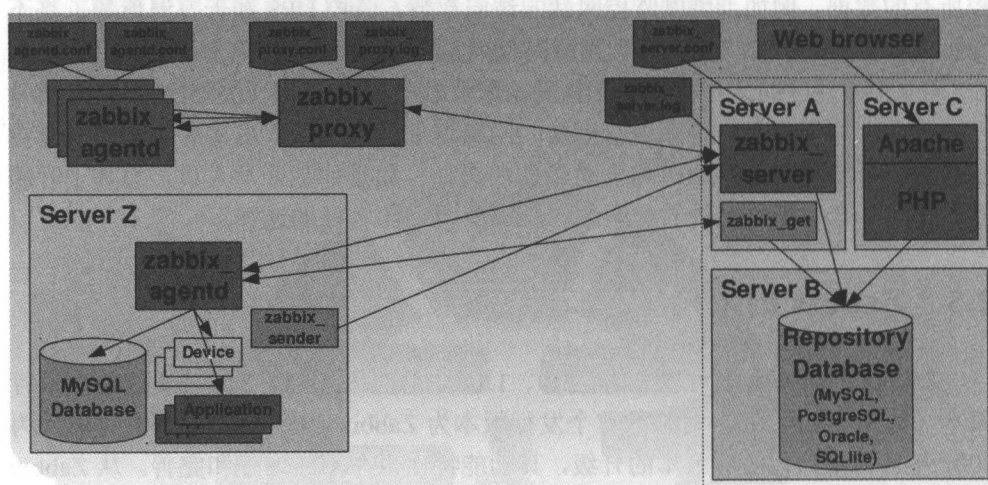


图 2-5

正如前面提到的，Zabbix 数据的采集不仅可以使用 Agent 方式，也可以使用其他方式，如 SNMP、SSH、Telnet、IPMI 等多种协议。

Zabbix-Server 的处理进程如图 2-6 所示。

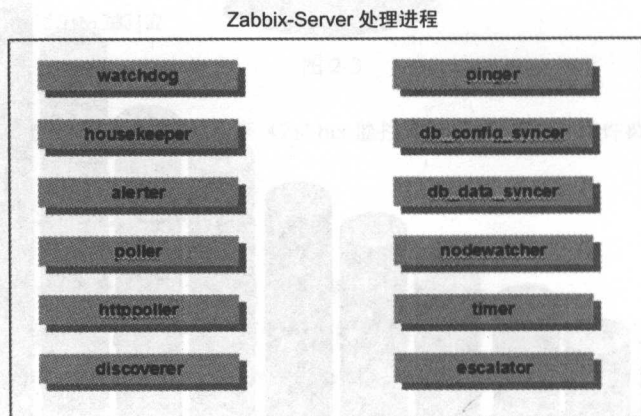


图 2-6

2.6 Zabbix 的运行流程

Zabbix 的运行流程可以用图 2-7 来表示。

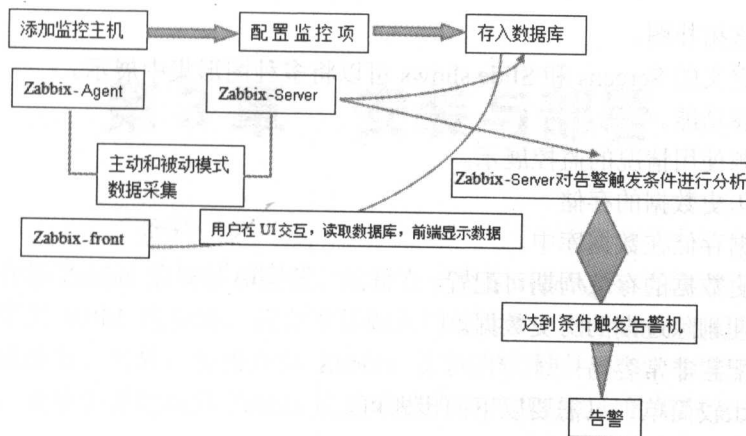


图 2-7

2.7 Zabbix 的功能特性

Zabbix 有常见的商业监控软件所具备的功能，如主机的性能监控、网络设备性能监控、数据库性能监控、FTP 等通用协议监控、多种告警方式、详细的报表图表绘制、分布式、可扩展能力、API 等。

(1) 数据收集

- 可用、性能检测。
- 支持 Agent、SNMP（包括 Trapping 和 Polling）、IPMI、JMX、SSH、Telnet 等。
- 自定义的检测。
- 自定义收集数据的频率。
- 服务器端/代理端和客户端模式。

(2) 灵活的触发器

可以定义非常灵活的告警阈值和多种告警相关联的条件。

(3) 高度可定制的告警

- 发送通知，可定制包括告警级别、动作升级、收件人和媒体类型。
- 通知可以使用全局宏变量和自定义的变量。
- 自动处理功能包括远程命令的自动调用和执行。

(4) 实时的绘图功能

监控项将数据实时绘制在图形上。

(5) Web 监控能力

Zabbix 可以模拟浏览器请求一个网站，并检查返回值和响应时间。

(6) 多种可视化的展示

- 可以自定义监控的展示图，将多种监控数据集中展示到一张图中。

- 网络拓扑图。
- 自定义的 Screens 和 Slide shows 可以将多种图形集中展示。
- 报表功能。
- 资源使用情况的监控展示。

(7) 历史数据的存储

- 数据存储于数据库中。
- 历史数据的存放周期可配置。
- 定期删除过期的历史数据。

(8) 配置非常容易

配置比较简单，只需要以下两步即可。

第一步：添加设备。

第二步：应用模板即可完成监控。

(9) 使用模板

- 模板可以分组。
- 模板具有可继承性。

(10) 网络发现

- 支持自动发现网络设备和服务器（可以通过配置自动发现服务规则实现）。
- Agent 自动注册。
- 支持自动发现（Low level discovery）实现动态监控项的批量监控（支持自定义），内置的自动发现包括文件系统、网络接口、SNMP OID，可定制自动发现。

(11) 快速的访问接口

- Web 页面基于 PHP。
- 远程访问。
- 日志审计。

(12) API 功能

应用 API 功能可以方便地和其他系统结合，包括手机客户端的使用。

(13) 系统权限

- 不同的用户展示监控的资源不同。
- 对用户的身份认证。

(14) 程序特性

用 C 语言编写，其性能和内存开销非常小。

(15) 大型环境的支持

利用 Zabbix-Proxy 方式即可轻松构建远程监控。

第 3 章 安装与部署

本章介绍 Zabbix 的安装和配置，这是在一个 Server/Agent 的架构环境中进行部署的，采用 RPM 包安装，适合零基础入门的读者。对于源码安装，请读者参考本书附录部分。另外，还将介绍 Zabbix 安装的软/硬件环境、容量规划等一些基础知识，对从头开始构建 Zabbix 监控系统的读者具有实际的指导意义。

3.1 安装环境概述

Zabbix 服务器运行的环境为 Linux(UNIX)+PHP+Web Service+DataBase，这里的 Web Service 可以为 Nginx、Apache，DataBase 可以为 MySQL、Oracle、PostgreSQL 等。

3.1.1 硬件条件

1. 硬件配置

官方推荐的最小硬件配置如表 3-1 所示。

表 3-1

环 境	平 台	CPU/内存	数 据 库	硬 盘	监控主机数量
小型	Ubuntu Linux	PII 350MHz 256MB	SQLite	普通	20
中型	Ubuntu Linux 64 bit	AMD Athlon 3200 2GB	MySQL InnoDB	普通	500
大型	Ubuntu Linux 64 bit	Intel Dual Core 6400 4GB	MySQL InnoDB 或 PostgreSQL	RAID10 SAS 或 SSD	>1000
超大型	RedHat Enterprise	Intel Xeon 2xCPU 8GB	MySQL InnoDB 或 PostgreSQL	RAID10 SAS 或 SSD	>10000

硬件需求与监控的机器数量和监控的数据量大小等有密切关系，对于硬盘的要求，建议采用 SAS 硬盘 RAID10，因为其性能和安全都很好。当然，实际环境中也有可能采用虚拟机来搭建 Zabbix 监控系统环境。在整个 Zabbix 监控系统的搭建和维护中，磁盘 I/O、数据库性能将成为整个监控系统运行良好的关键因素。

2. 磁盘容量大小

Zabbix-Server 的数据库大小取决于 NVPS (Number of processed values per

second)，如图 3-1 所示，NVPS 从总体上反映了处理速度，与监控项的数目、监控的类型、取值间隔、History 的保留时间和 Trends 的保留时间有直接关系。

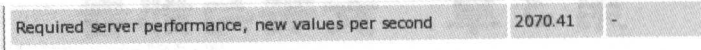


图 3-1

(1) 历史数据的保存

NVPS 值是每秒处理的平均数量，假设有 60000 个监控项，刷新周期为 60 秒，那么每秒需要处理的数据为 $60000/60=1000$ （条），表示此时每秒将会向数据库中新增 1000 条数据，这些数据根据设置的历史存储时间保存。例如，保存周期为 90 天，每秒处理 1000 条，计算方法为：

历史数据大小=天数×每秒处理的数据×一天 24 小时×一小时 3600 秒×50Bytes

则需要占用的空间为：

$(90 \times 24 \times 3600) \times 1000 \times 50 = 388\,800\,000\,000$ (B) (约 362GB，假设保存为一年，则有 $362 \times 4 = 1448$ GB)

(2) 趋势数据的保存

每一个 Items 趋势的数据大约为 128B，这取决于数据库类型。

假设有 60 000 个 Items，保存一年的趋势数据，其计算方法为：

趋势数据大小=监控项×每天 24 小时×一年 365 天×128B

则需要的空间为：

$60\,000 \times 24 \times 365 \times 128 = 67\,276\,800\,000$ B (约 67GB)

(3) 事件数据的保存

每个事件占用 130 B 空间，假设保存一年，则需要的空间为：

$1 \times 24 \times 3600 \times 365 \times 130 = 4\,099\,680\,000$ B (约 4GB)

即 60 000 个监控项，周期为 60 秒更新数据，保存一年需要 $362+67+4=433$ (GB) 磁盘空间，实际上，比这占用得更多，需要保存其他更多的数据，因此，应该会超过 500GB 的数据空间。

(4) 需要的空间计算方法

磁盘空间的计算方法如表 3-2 所示。

表 3-2

选 项	需要的磁盘空间 (单位 B)
Zabbix 配置数据	约 10MB
历史数据	$\text{days} \times (\text{items} / \text{刷新时间}) \times 24 \times 3600 \times \text{bytes}$ items: item 的数量 days: 历史天数 refresh rate: 监控项的平均取值周期 bytes: 单个值的大小，取决于数据库引擎，通常为 50B

续表

选 项	需要的磁盘空间 (单位 B)
历史趋势数据	$\text{days} \times (\text{items} / 3600) \times 24 \times 3600 \times \text{bytes}$ items: items 的数量 days: 历史数据的天数 bytes: 取决于数据库引擎, 通常为 128 B
事件数据	$\text{days} \times \text{events} \times 24 \times 3600 \times \text{bytes}$ events: 每秒的事件数量 days: 历史数据的天数 bytes: 取决于数据库引擎, 通常为 130 B

所以, 总共需要的磁盘空间大小计算公式为:

$$\text{Total space} = \text{Configuration} + \text{History} + \text{Trends} + \text{Events}$$

可以看到, 由于 Items、取值间隔、历史数据等各因素的不同, 将会使磁盘空间的使用量不同。对数据库来说, 巨大的数据量会造成数据库查询的压力增大, 这将在后面探讨。因此, 对监控系统进行规划之前, 必须要考虑到硬盘性能、监控数据的存放周期等因素。

3.1.2 软件条件

1. Zabbix 支持的操作系统平台

Zabbix 支持 Linux、UNIX、Windows 系统, 如表 3-3 所示, 列出了 Zabbix 各模块所能支持的操作系统。

表 3-3

系统发行版	Zabbix-Server	Zabbix-Proxy	Zabbix-Agent
AIX	支持	支持	支持
FreeBSD	支持	支持	支持
HP-UX	支持	支持	支持
Linux	支持	支持	支持
Mac OS X	支持	支持	支持
Novell Netware	不支持	不支持	支持
Open BSD	支持	支持	支持
SCO Open Server	支持	支持	支持
Solaris	支持	支持	支持
Tru64/OSF	支持	支持	支持
Windows NT 4.0、Windows 2000、Windows Server 2003、Windows XP、Windows Vista、Windows Server 2008、Windows 7、Windows Server 2012、Windows 8	不支持	不支持	支持

注意：如果仅仅检测网络服务 FTP、SSH、HTTP、DNS、LDAP，无须安装任何客户端，即可支持监控数据的获取。

2. 数据库版本的要求

数据库版本的要求如表 3-4 所示。

表 3-4

数据库名称	要求版本（含自身）	备注信息
MySQL	5.0.3 以上	需支持 InnoDB 存储引擎
Oracle	10g 以上	无其他要求，可以在线使用
PostgreSQL	8.1 以上	建议使用 PostgreSQL 8.3，它拥有更好的性能
SQLite	3.3.5 以上	无其他要求，一般 10 台机器以下采用，常用于测试
IBM DB2	9.7 以上	无其他要求，用于支持实验

3. Web 前端需要支持的软件环境

Web 前端需要支持的软件环境如表 3-5 所示。

表 3-5

软件	版本	备注
Apache	1.3.12 以上	
PHP	5.3.0 以上	
PHP 扩展库支持		
gd	2.0 以上	PHP GD 扩展需支持 PNG (<code>--with-png-dir</code>)、JPEG (<code>--with-jpeg-dir</code>) 和 FreeType 2 (<code>--with-freetype-dir</code>)
bcmath		php-bcmath (<code>--enable-bcmath</code>)
ctype		php-ctype (<code>--enable-ctype</code>)
libXML	2.6.15 以上	php-xml 或者 php5-dom
xmlreader		php-xmlreader
Xmlwriter		php-xmlwriter
session		php-session
sockets		php-net-socket (<code>--enable-sockets</code>)
mbstring		php-mbstring (<code>--enable-mbstring</code>)
gettext		php-gettext (<code>--with-gettext</code>)
ibm_db2		使用 DB2 需此支持
mysqli		使用 MySQL 需此支持
oci8		使用 Oracle 需此参数
pgsql		使用 PostgreSQL 需此支持
sqlite3		使用 SQLite 需此支持

如果是 RHEL 系统，系统自带的 RPM 包会缺少 php-mbstring 和 php-bcmath 两个包，这个问题后面有介绍。

如果是编译安装 PHP，请确保以上参数开启。

4. 用户浏览器

浏览器需要支持 Cookies、JavaScript，常见的浏览器都可以支持。

5. Zabbix-Server 需要的软件环境

Zabbix-Server 需要的软件环境如表 3-6 所示。

表 3-6

依赖的软件包	描 述
OpenIPMI	如需支持 IPMI
libssh 2	如需支持 SSH，则需要 libssh 1.0 或更高版本
fping	ICMP ping 的支持
libcurl	Web 监控
libksemel	Jabber 告警介质
net-snmp	SNMP 的支持

6. 时间同步

Zabbix-Server 对时间的精准要求比较高，时间对数据的计算等都有影响，因此，最好设置 ntp 自动同步时间。也可以用 crontab 进行同步（在实际的生产环境中不推荐这么做），如下所示，使用 crontab 进行时间同步。

```
* /30 * * * * /usr/sbin/ntpdate pool.ntp.org
```

3.1.3 部署环境的考虑

Zabbix-Server 尽量部署在核心业务所在的机房，要求到各分节点的网络稳定。另外，要考虑到网络流量问题，做好安全策略。

3.2 Zabbix-Server 服务器端的安装

下面以 CentOS6.4_X64 为例介绍如何安装 Zabbix-Server 服务器端。本例采用 RPM 包安装方式，这也是本书推荐的方式，该方式较简捷。源码安装的方式可以参考本书附录部分。

准备好 CentOS 6.4_X64 系统，配置 IP 地址，确保与互联网连接正常。
进入系统测试网络（如图 3-2 所示）。

```
[root@zabbix ~]# ping -c 4 www.google.com
PING www.google.com (74.125.31.99) 56(84) bytes of data.
64 bytes from tb-in-f99.1e100.net (74.125.31.99): icmp_seq=1 ttl=45 time=62.2 ms
64 bytes from tb-in-f99.1e100.net (74.125.31.99): icmp_seq=2 ttl=45 time=65.7 ms
64 bytes from tb-in-f99.1e100.net (74.125.31.99): icmp_seq=3 ttl=45 time=67.2 ms
64 bytes from tb-in-f99.1e100.net (74.125.31.99): icmp_seq=4 ttl=45 time=64.9 ms
```

图 3-2

安装 Zabbix 官方源和 epel 源, 如图 3-3 所示。

```
shell# rpm -ivh http://repo.zabbix.com/zabbix/2.2/rhel/6/x86_64/zabbix-release-2.2-1.el6.noarch.rpm
shell# rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

注意: 由于软件版本的更新, 以上链接可能会失效, 请读者下载并安装相应的版本。

```
[root@zabbix ~]# rpm -ivh http://repo.zabbix.com/zabbix/2.2/rhel/6/x86_64/zabbix-release-2.2-1.el6.noarch.rpm
Retrieving http://repo.zabbix.com/zabbix/2.2/rhel/6/x86_64/zabbix-release-2.2-1.el6.noarch.rpm
warning: /var/tmp/rpm-tmp.AFNMea: Header V4 DSA/SHA1 Signature, key ID 79eae5d3: NOKEY
Preparing... [100%]
1:zabbix-release
[root@zabbix ~]# rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
Retrieving http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
warning: /var/tmp/rpm-tmp.McA8GT: Header V3 RSA/SHA256 Signature, key ID 0608b895: NOKEY
Preparing... [100%]
1:epel-release
[root@zabbix ~]#
```

图 3-3

3.2.1 安装 Zabbix-Server

安装服务器端 (注意: 由于服务器端也是需要监控的, 故这里也一并安装 Zabbix-Agent, 如图 3-4 所示)。

```
shell# yum install -y zabbix zabbix-get zabbix-server zabbix-web-mysql zabbix-web zabbix-agent
```

```
Installed:
  zabbix.x86_64 0:2.2.0-1.el6      zabbix-agent.x86_64 0:2.2.0-1.el6      zabbix-get.x86_64 0:2.2.0-1.el6      zabbix-server.x86_64 0:2.2.0-1.el6
  zabbix-web.noarch 0:2.2.0-1.el6  zabbix-web-mysql.noarch 0:2.2.0-1.el6

Dependency Installed:
  epel.noarch 0:2.2.0-1.el6      apr.x86_64 0:1.3.9-1.el6.2      apr-util.x86_64 0:1.3.9-1.el6.0.1
  openssl-libs.x86_64 0:1.0.1e-14.el6      dejavu-fonts-common.noarch 0:2.30-2.el6
  perl-libs.x86_64 0:5.10.0-1.el6      freetype.x86_64 0:2.3.11-14.el6.3.1
  fontpackages-filesystem.noarch 0:1.41-1.1.el6      httpd.x86_64 0:2.2.15-29.el6.centos
  glibc.x86_64 0:2.12-100.el6.4.2      libX11.x86_64 0:1.3.0-4.el6
  libXext.x86_64 0:1.3.0-2.el6          libXp.x86_64 0:1.0.3-101.el6
  libXft.x86_64 0:2.1.2-49.el6.2        libbrotli.x86_64 0:1.0.2-1.el6
  libXrender.x86_64 0:0.3.6-1.el6        libcurl.noarch 0:7.19.1-2.el6
  libxml2.x86_64 0:2.7.6-13.5.el6        perl.x86_64 4:5.10.0-131.el6.4
  net-snmp-libs.x86_64 1:5.3.44.el6.2      perl-libs.x86_64 4:5.10.0-131.el6.4
  perl-Pod-escape.x86_64 1:1.04-11.el6.4    php.x86_64 0:5.3.3-27.el6.3
  perl-libs.noarch 0:5.10.0-131.el6.4      php-common.x86_64 0:5.3.3-27.el6.3
  php-cli.x86_64 0:5.3.3-27.el6.3          php-mysql.x86_64 0:5.3.3-27.el6.3
  php-mysql.noarch 0:5.3.3-27.el6.3        unixodbc.x86_64 0:2.2.14-12.el6.1
  php-xml.x86_64 0:5.3.3-27.el6.3          zabbix-server-mysql.x86_64 0:2.2.0-1.el6
```

图 3-4

安装完成后可以看到, yum 方式安装并没有自动安装 mysql-server。

注意: 由于 RHEL 系统 (CentOS 不存在这个问题) 缺少 php-bcmath 和 php-mbstring 这两个包, 会导致 Web 页面的安装提示缺少组件, 所以读者可以在 CentOS 源中下载这两个包安装即可 (注意与系统版本对应)。

<http://vault.centos.org/>

3.2.2 安装 MySQL 数据库服务

安装 MySQL 数据库服务，命令如下：

```
shell# yum -y install mysql-server
```

所需的依赖包如图 3-5 所示。

```
Running Transaction
  Updating      : mysql-libs-5.1.69-1.el6_4.x86_64
  Installing    : perl-DBI-1.609-4.el6.x86_64
  Installing    : perl-DBD-MySQL-4.013-3.el6.x86_64
  Installing    : mysql-5.1.69-1.el6_4.x86_64
  Installing    : mysql-server-5.1.69-1.el6_4.x86_64
  Cleanup      : mysql-libs-5.1.66-2.el6_3.x86_64
  Verifying     : mysql-libs-5.1.69-1.el6_4.x86_64
  Verifying     : perl-DBD-MySQL-4.013-3.el6.x86_64
  Verifying     : perl-DBI-1.609-4.el6.x86_64
  Verifying     : mysql-server-5.1.69-1.el6_4.x86_64
  Verifying     : mysql-5.1.69-1.el6_4.x86_64
  Verifying     : mysql-libs-5.1.66-2.el6_3.x86_64
```

图 3-5

修改 MySQL 配置文件如下（黑体字部分很重要）。

```
shell# vi /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
character-set-server=utf8 #设置字符集为utf8
innodb_file_per_table=1  #让innodb的每个表文件单独存储

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

启动服务。

```
shell# chkconfig mysqld on
shell# service mysqld start
```

查看服务是否启动成功，如图 3-6 所示。

```
[root@zabbix ~]# ps -ef |grep mysql
root      2006      1  0 18:47 pts/0    00:00:00 /bin/sh /usr/bin/mysqld
mysql     2114    2006      0 18:47 pts/0    00:00:00 /usr/libexec/mysqld
root      2131    1741      0 18:48 pts/0    00:00:00 grep mysql
[root@zabbix ~]# netstat -nlp |grep 3306
tcp       0      0 0.0.0.0:3306  0.0.0.0:*
```

图 3-6

1. 创建 Zabbix 的数据库

设置 MySQL 的 root 用户密码。


```

shell# mysqladmin -uroot password admin
shell# mysql -uroot -padmin
mysql> create database zabbix character set utf8;
mysql> grant all privileges on zabbix.* to zabbix@localhost id
entified by 'zabbix';
mysql> flush privileges;

```

注意：这里容易出现的问题是创建 Zabbix 的数据库字符集不为 utf8，这会导
致 Web 界面切换到中文环境时出现乱码。

数据库不为 utf8，出现中文乱码的界面如图 3-7 所示。

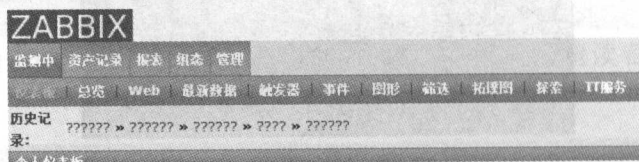


图 3-7

2. 导入 Zabbix-Server 的数据库

```

shell# mysql -uzabbix -pzabbix
mysql> use zabbix
mysql> source /usr/share/doc/zabbix-server-mysql-X.X.X/create/sc
hema.sql;

```

需要注意的是，如果安装 Zabbix-Proxy，只导入 schema.sql 即可，无须导入
下面的 SQL，否则 Zabbix-Proxy 无法正常工作。

```

mysql> source /usr/share/doc/zabbix-server-mysql-X.X.X/create/im
ages.sql;
mysql> source /usr/share/doc/zabbix-server-mysql-X.X.X/create/da
ta.sql;
mysql> show tables;

```

3.2.3 配置 zabbix_server.conf

配置 zabbix_server.conf 文件如下。

1. 默认参数

```

shell# egrep -v "(^#|^$)" /etc/zabbix/zabbix_server.conf
LogFile=/var/log/zabbix/zabbix_server.log
LogFileSize=0
PidFile=/var/run/zabbix/zabbix_server.pid
DBName=zabbix
DBUser=zabbix
DBSocket=/var/lib/mysql/mysql.sock
SNMPTrapperFile=/var/log/snmp/snmpd.log
AlertScriptsPath=/usr/lib/zabbix/alertscripts
ExternalScripts=/usr/lib/zabbix/externalscripts

```

2. 修改后的参数 (可参考)

```

shell# egrep -v "(^#|^$)" /etc/zabbix/zabbix_server.conf
LogFile=/var/log/zabbix/zabbix_server.log
LogFileSize=0
PidFile=/var/run/zabbix/zabbix_server.pid
DBHost=localhost                #可修改
DBName=zabbix                   #默认
DBUser=zabbix                   #默认
DBPassword=zabbix               #需修改
DBSocket=/var/lib/mysql/mysql.sock #默认
DBPort=3306                     #默认
StartPollers=5
StartIPMIPollers=10
StartPollersUnreachable=10
StartTrappers=10
StartPingers=10
StartDiscoverers=10
VMwareFrequency=60
VMwareCacheSize=8M
SNMPTrapperFile=/var/log/snmpd/snmpd.log
ListenIP=127.0.0.1
MaxHousekeeperDelete=500
CacheSize=256M
StartDBSyncers=40
HistoryCacheSize=128M
TrendCacheSize=128M
HistoryTextCacheSize=128M
ValueCacheSize=128M
Timeout=30
TrapperTimeout=300
UnreachablePeriod=45
UnavailableDelay=60
UnreachableDelay=15
AlertScriptsPath=/etc/zabbix/alertscripts
ExternalScripts=/etc/zabbix/externalscripts
FpingLocation=/usr/sbin/fping
LogSlowQueries=10000
StartProxyPollers=50
ProxyConfigFrequency=3600

```

以上参数只需关注**黑体字**部分, 这部分为性能参数, 需根据实际情况进行调整。默认只需修改 **DBPassword=zabbix** 即可。

```
shell# mkdir /etc/zabbix/alertscripts /etc/zabbix/externalscripts
```

3. 启动 Zabbix-Server 服务

```

shell# service zabbix-server start
Starting Zabbix server: [ OK ]
shell# service httpd start
Starting httpd: [ OK ]

```

添加开机启动项。

```
shell# chkconfig zabbix-server on
shell# chkconfig httpd on
```

3.2.4 防火墙、Selinux 和权限的设置

1. 防火墙的设置

```
shell# vim /etc/sysconfig/iptables
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 10051 -j ACCEPT
-A OUTPUT -m state --state NEW -m tcp -p tcp --dport 10050 -j ACCEPT
#-A INPUT -m state --state NEW -m tcp -p tcp --sport 10050 -j ACCEPT
shell# service iptables restart
```

上述代码中，10050 是 Agent 的端口，Agent 采用被动方式，Server 主动连接 Agent 的 10050 端口；10051 是 Server 的端口，Agent 采用主动或 Trapper 方式，会连接 Server 的 10051 端口。

2. Selinux 的设置

如果开启 Selinux，安装时提示不能写入文件，如图 3-8 所示，除文件权限不是 Web 用户外，还有一个原因是 Selinux 默认不允许写入文件，设置语句如下：

```
shell# chcon -R -t httpd_sys_content_rw_t /usr/share/zabbix/conf
shell# setsebool -P httpd_can_network_connect=true
shell# semanage port -a -t http_port_t -p tcp 10051
```

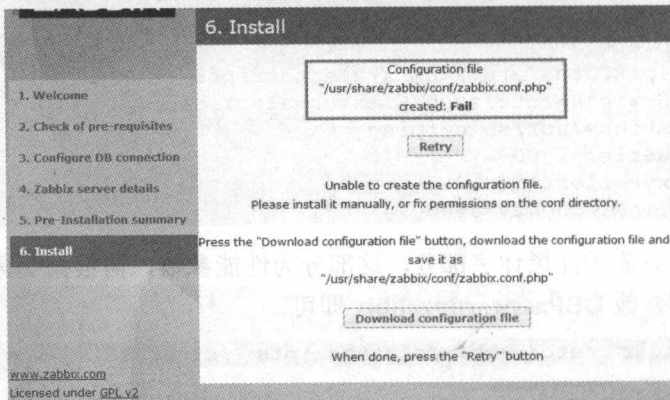


图 3-8

如果读者对 Selinux 不是特别熟悉，可以选择关闭 Selinux。在实际的生产环境中，大部分用户也是选择关闭 Selinux 的。

```
shell# setenforce 0 #设置为警告模式，只给出提示，不会阻止操作，不用重启
                        #服务器即生效
shell# getenforce    #获取当前Selinux的运行状态
                        【Enforcing|Permissive|Disabled】
```

直接关闭 Selinux 的方法。

```
shell# vim /etc/selinux/config
SELINUX=disabled
```

注意：此种方式需要重启服务器才能生效。

3. php.ini 配置文件的设置

```
shell# vim /etc/php.ini
date.timezone = Asia/Shanghai
max_execution_time = 300
post_max_size = 16M
max_input_time=300
memory_limit = 128M
mbstring.func_overload = 2
```

在 LAMP 环境中，也可以按上述方式配置 PHP 的参数，比修改 php.ini 更方便。而在 Zabbix 的官方 RPM 中，这一步已经配置过了，所以无须修改。

```
shell# vim /etc/httpd/conf.d/zabbix.conf
<Directory "/usr/share/zabbix">
    Options FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
    php_value date.timezone Asia/Shanghai
    php_value max_execution_time 300
    php_value post_max_size 16M
    php_value max_input_time 300
    php_value memory_limit 128M
    php_value upload_max_filesize 2M
</Directory>
shell# service httpd restart
```

如果在后面配置 Web 时提示任何参数不满足安装配置要求（如图 3-9 所示），修改对应的参数后重启 Httpd 即可。

	Current value	Required	
PHP version	5.3.3	5.3.0	OK
PHP option memory_limit	128M	128M	OK
PHP option post_max_size	16M	16M	OK
PHP option upload_max_filesize	2M	2M	OK
PHP option max_execution_time	300	300	OK
PHP option max_input_time	300	300	OK
PHP time zone	unknown		Fail

图 3-9

3.2.5 配置 Web 界面

打开浏览器，输入“http://IP 地址/zabbix”，会出现如图 3-10 所示的界面。

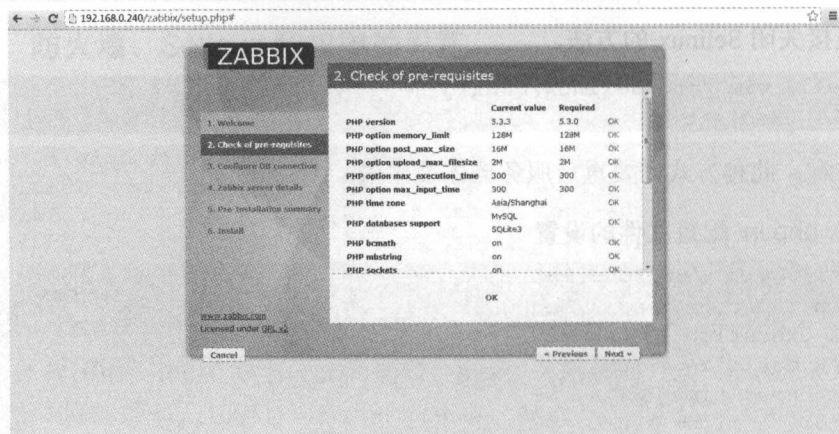


图 3-10

PHP 提示的各项参数都满足后，可以继续往下进行，单击“Next”按钮。如果提示参数不通过，修改 php.ini 配置文件，并重启 Web (httpd) 服务。

配置数据库连接的各项参数如图 3-11 所示。

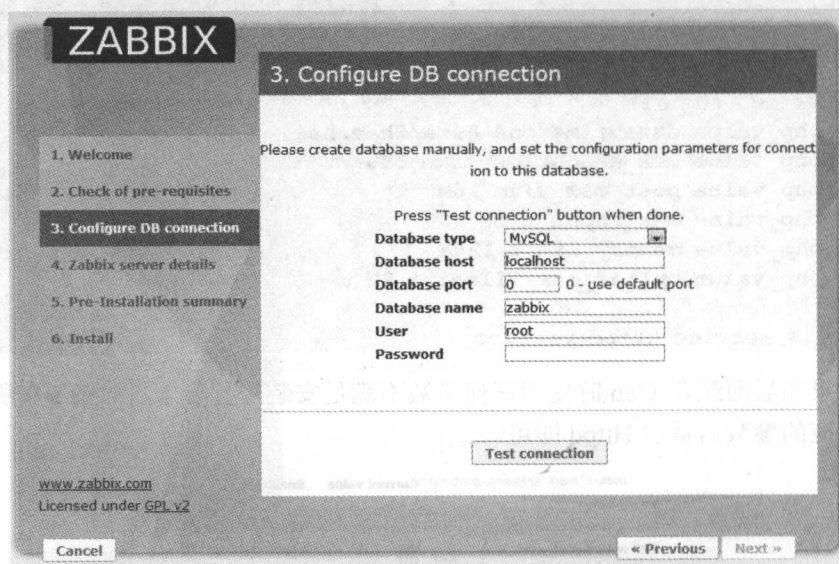


图 3-11

单击“Test connection”按钮测试数据库是否正常连接，如图 3-12 所示。

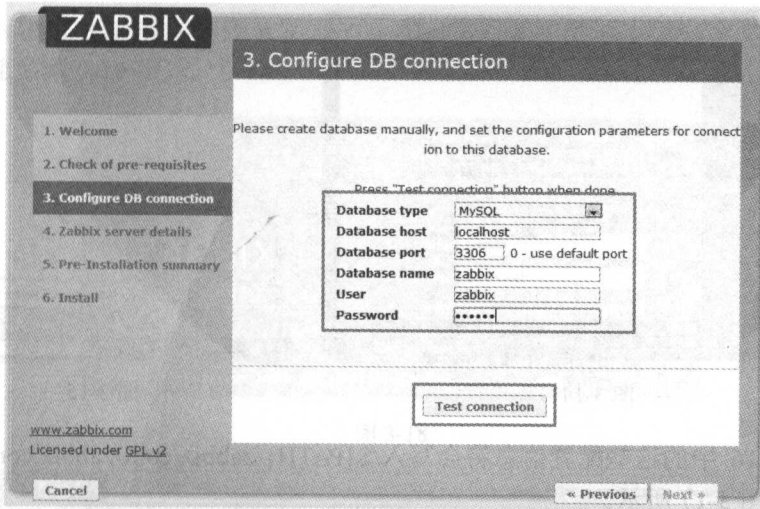


图 3-12

测试结果为 OK 后，单击“Next”按钮进行下一步操作，如图 3-13 所示。

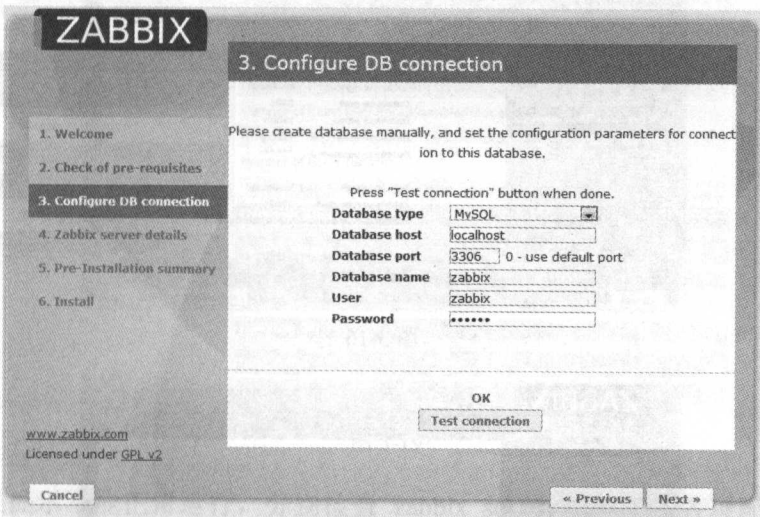


图 3-13

Zabbix-Server 的链接地址、端口、名称设置（见图 3-14）将会显示在 Zabbix 的前端页面，如图 3-15 所示。

注意：如果 Zabbix-Server 在其他机器中，这里的 Host 填写 Zabbix-Server 所在机器的 IP。

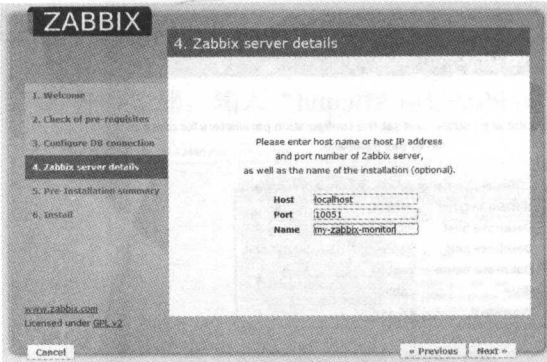


图 3-14

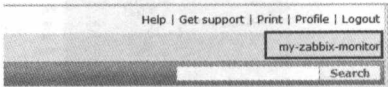


图 3-15

图 3-16 中的这些配置信息将会写入 `/${PATH}/zabbix/conf/zabbix.conf.php` 文件中，如图 3-17 所示。

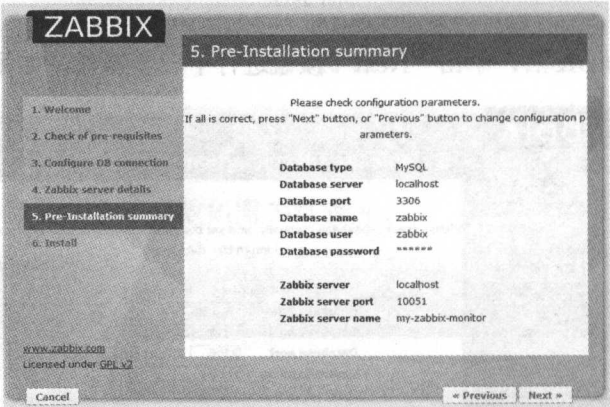


图 3-16

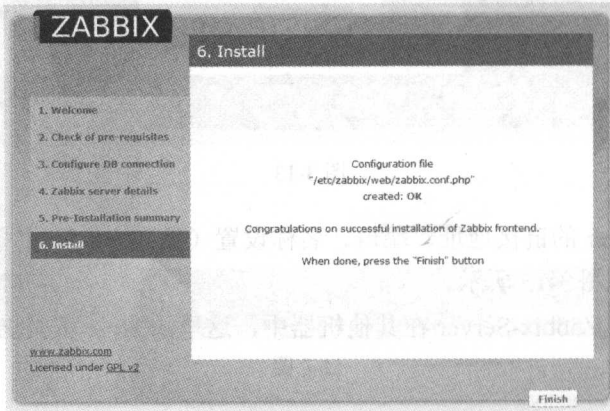


图 3-17

进入如图 3-18 所示的登录界面，默认账户是 Admin，密码是 zabbix，登录成功的页面如图 3-19 所示。

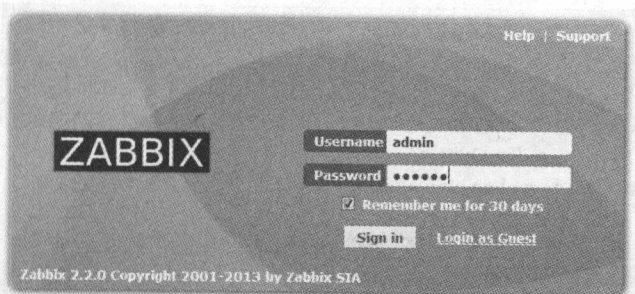


图 3-18

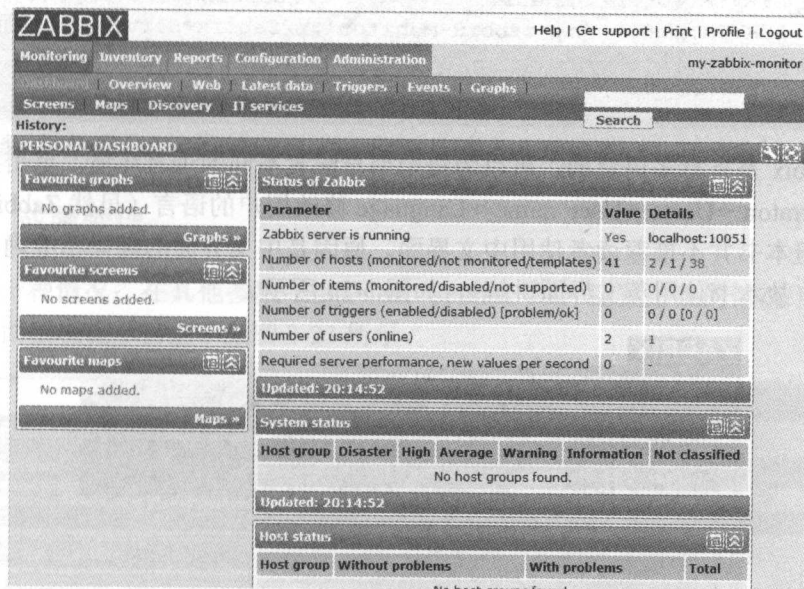


图 3-19

如果提示不能正常运行，分别检查 `zabbix_server.conf` 中的各项配置文件、Selinux、Iptables 等。

配置文件请检查以下文件的正确配置参数。

① `/etc/zabbix/zabbix_server.conf` 中的参数。

<code>DBHost = X.X.X.X</code>	#数据库的IP (域名) 地址
<code>DBName=zabbix</code>	#数据库的名称
<code>DBUser=zabbix</code>	#数据库的用户
<code>DBPassword=zabbix</code>	#数据库的密码

② `/usr/share/zabbix/conf/zabbix.conf.php` 中的配置。

```

<?php
// Zabbix GUI configuration file
global $DB;

$DB['TYPE']      = 'MYSQL';           //数据库类型
$DB['SERVER']    = 'mysql.itnihao.com'; //数据库的IP (域名) 地址
$DB['PORT']      = '3306';           //数据库的端口
$DB['DATABASE']  = 'zabbix';         //数据库的名称
$DB['USER']      = 'zabbix';         //数据库的用户
$DB['PASSWORD']  = 'zabbix';         //数据库的密码

// SCHEMA is relevant only for IBM_DB2 database
$DB['SCHEMA'] = '';

$ZBX_SERVER      = 'zabbix.itnihao.com'; //Zabbix-Server的IP (域名) 地址
$ZBX_SERVER_PORT = '10051';             //Zabbix-Server的端口
$ZBX_SERVER_NAME = 'my-zabbix-monitor'; //Zabbix-Server web界面的标识

$IMAGE_FORMAT_DEFAULT = IMAGE_FORMAT_PNG;
?>

```

Zabbix 是支持多语言的，可以为每个用户配置不同的语言环境。单击菜单栏 Administration→Users→User name→Language 修改用户的语言（虽然 Zabbix 支持中文，但本书并不推荐读者使用中文界面，原因是中文界面的翻译不准确，会误导读者，故本书使用英文界面讲解），如图 3-20 所示。

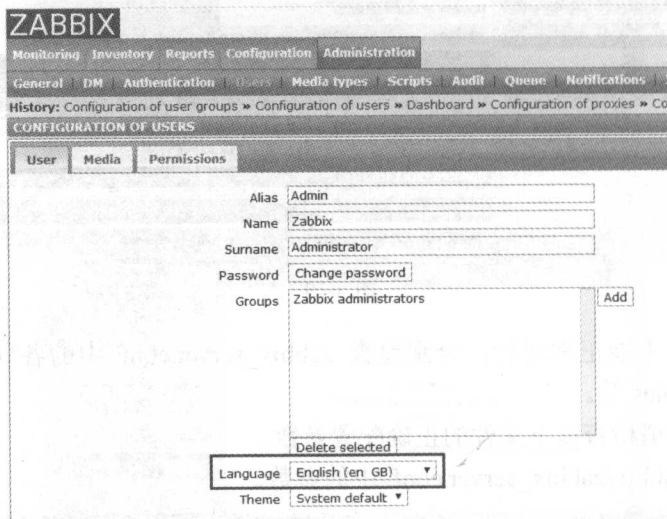


图 3-20

另外，在个人用户设置中（见图 3-21）可以开启声音告警的提示信息 and 前端消息的声音提示（见图 3-22）。

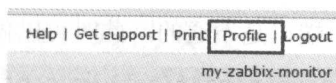


图 3-21

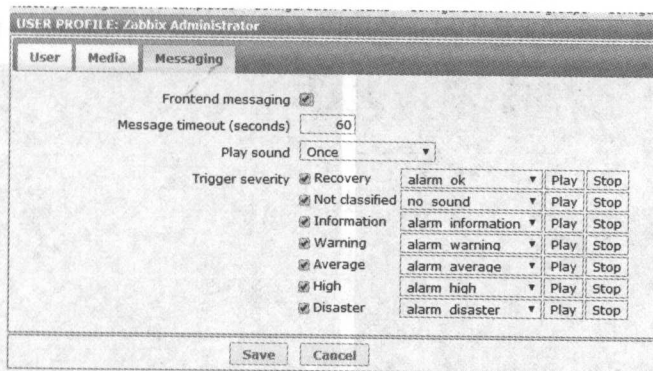


图 3-22

安装成功后，请务必禁用 Guests 账户，防止非法访问。

3.2.6 故障处理

第一种情况：在其他参数（Iptables、Selinux 等）配置正确的情况下，如果 Web 界面出现提示信息，如图 3-23 所示。

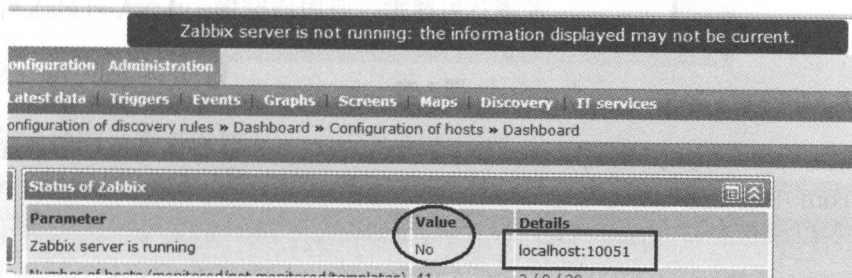


图 3-23

提示无法连接，则进入 Zabbix-Server 所在的系统，利用 Telnet 进行测试，如图 3-24 所示。

```
[root@localhost ~]# telnet localhost 10051
telnet: localhost: Temporary failure in name resolution
localhost: Host name lookup failure
[root@localhost ~]# telnet 127.0.0.1 10051
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^['.
```

图 3-24

看到 localhost 的 10051 端口是不通的，此时可以做如下调整。

```
shell# vim ${PATH}/zabbix/conf/zabbix.conf.php # (如图3-25所示)
(/usr/share/zabbix/conf/zabbix.conf.php#官方RPM安装的PHP代码文件位置)
```

将 localhost 修改为 127.0.0.1，修改图 3-25 的内容为图 3-26。

```
// Zabbix GUI configuration file
global $DB;

$DB['TYPE'] = 'MYSQL';
$DB['SERVER'] = 'localhost';
$DB['PORT'] = '3306';
$DB['DATABASE'] = 'zabbix';
$DB['USER'] = 'zabbix';
$DB['PASSWORD'] = 'zabbix';

// SCHEMA is relevant only for IBM_DB2 database
$DB['SCHEMA'] = '';

$ZBX_SERVER = 'localhost';
$ZBX_SERVER_PORT = '10051';
$ZBX_SERVER_NAME = 'Zabbix Server';

$IMAGE_FORMAT_DEFAULT = IMAGE_FORMAT_PNG;
```

图 3-25

```
// Zabbix GUI configuration file
global $DB;

$DB['TYPE'] = 'MYSQL';
$DB['SERVER'] = '127.0.0.1';
$DB['PORT'] = '3306';
$DB['DATABASE'] = 'zabbix';
$DB['USER'] = 'zabbix';
$DB['PASSWORD'] = 'zabbix';

// SCHEMA is relevant only for IBM_DB2 database
$DB['SCHEMA'] = '';

$ZBX_SERVER = '127.0.0.1';
$ZBX_SERVER_PORT = '10051';
$ZBX_SERVER_NAME = 'Zabbix Server';

$IMAGE_FORMAT_DEFAULT = IMAGE_FORMAT_PNG;
```

图 3-26

第二种情况：数据库 mysql.sock 文件无法找到的问题（见图 3-27）。

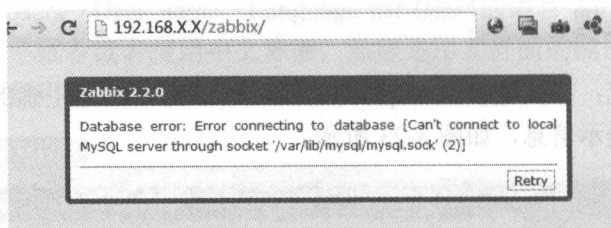


图 3-27

确保/var/lib/mysql/mysql.sock 存在，如果不存在，请修改/etc/zabbix/zabbix_server.conf 中的 DBSocket 配置。

```
DBSocket=/var/lib/mysql/mysql.sock
```

注意：修改为 mysql.sock 实际存在的路径后，重启服务。

第三种情况：数据库无法连接提示（见图 3-28）。

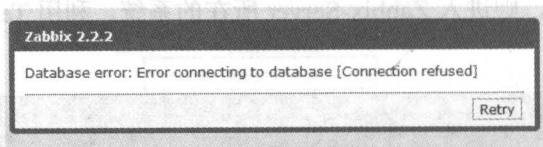


图 3-28

解决方法有三种：检查数据库服务是否正常；防火墙是否允许；权限能否访问。

3.3 Zabbix-Agent 客户端的安装

3.3.1 安装 Zabbix-Agent

前面我们已经知道了 Zabbix 可以采用 Agent/Server 的架构, 客户端的采集方式分为 Agent、SNMP 等, 这里先学习用 Agent 方式对服务器进行监控, 在第 6 章中, 将对其他监控方式 (SNMP、JMX、IPMI) 进行详细介绍, 下面介绍 Zabbix-Agent 手动安装的方式。

这里的客户端依然采用 RPM 方式安装 (安装 Zabbix 官方的 yum 源)。

```
shell# rpm -ivh http://repo.zabbix.com/zabbix/2.2/rhel/6/x86_64/  
/zabbix-release-2.2-1.el6.noarch.rpm  
shell# yum install -y zabbix zabbix-agent
```

注意: Zabbix 官方的 RPM 包会发布新版本, 所以读者可以选择用 yum 安装最新版本的 RPM 包。

这种安装方式比较简单, 假如要对 1000 台服务器进行监控, 并且采用 Agent 方式, 就需要在 1000 台服务器上安装 Zabbix-Agent, 对于这种规模的 Agent 安装, 采用自动化配置管理工具无疑是必须选择的方式, 并且需要对 Zabbix-Agent 进行软件包的定制, 如将自定义的参数、配置文件等放在 RPM 包中。关于 RPM 包的定制, 请读者参考第 15 章。

由于我们需要对 Zabbix-Server 进行监控, 所以在 Zabbix-Server 服务器中也需要安装 Zabbix-Agent, 安装完毕后, 需要进行配置。

3.3.2 防火墙的设置

防火墙的设置语句如下。

```
shell# vi /etc/sysconfig/iptables  
-A INPUT -m state --state NEW -m tcp -p tcp --dport 10050 -j ACCEPT  
-A OUTPUT -m state --state NEW -m tcp -p tcp --dport 10051 -j ACCEPT  
shell# service iptables restart
```

3.3.3 配置 zabbix_agentd.conf

默认的配置参数如下。

```
shell# egrep -v "(^#|^$)" /etc/zabbix/zabbix_agentd.conf  
PidFile=/var/run/zabbix/zabbix_agentd.pid  
LogFile=/var/log/zabbix/zabbix_agentd.log  
LogFileSize=0  
Server=127.0.0.1  
ServerActive=127.0.0.1  
Hostname=Zabbix server  
Include=/etc/zabbix/zabbix_agentd.d/
```


需改变的参数如下。

Server: 被动模式, 允许哪台服务器连接Agent。

ServerActive: 主动模式, 向哪台服务器传送数据。

关于主动和被动这两种 Agent 的工作模式, 请读者参考第 7 章的内容。

一个 Agent 是可以同时向多个服务器端发送数据的, 多个 IP 用逗号分隔。

Server=127.0.0.1,192.168.0.240, 表示 Server 的 IP 为 127.0.0.1 和 192.168.0.240, 这两台 Zabbix-Server 服务器端可获取此 Agent 端的监控数据。

```
shell# egrep -v "(^#|^$)" /etc/zabbix/zabbix_agentd.conf
PidFile=/var/run/zabbix/zabbix_agentd.pid
LogFile=/var/log/zabbix/zabbix_agentd.log
LogFileSize=0
Server=127.0.0.1,192.168.0.240           #被动模式, Zabbix-Server的IP地址
ServerActive=192.168.0.240:10051       #主动模式
Hostname=Zabbix server
Include=/etc/zabbix/zabbix_agentd.d/
UnsafeUserParameters=1
shell# chkconfig zabbix-agent on
shell# service zabbix-agent start
Starting Zabbix agent: [ OK ]
```

3.4 SNMP 监控方式的配置

由于某些设备并不能安装 Agent, 或者处于安装 Agent 不方便的因素考虑, 将采用 SNMP 来监控。本节只介绍 Linux 下 SNMP 的配置。关于 SNMP 的原理、监控配置的知识, 请参考第 6 章。

下面配置 Linux 下 SNMP 的监控, 语句如下。

```
shell# yum -y install net-snmp
shell# vim /etc/snmp/snmpd.conf
com2sec mynetwork 192.168.0.240 public_monitor
com2sec mynetwork 127.0.0.1 public
group MyROGroup v2c mynetwork
access MyROGroup "" any noauth prefix all none none
view all included .1 80
shell# chkconfig snmpd on
shell# service snmpd restart
```

3.5 在 Windows 中安装 Zabbix-Agent

1. 安装配置

下载 Zabbix-Agent 的 Windows 版本, 这里以 Zabbix-Agent 2.2.0 为例, 下载地址为 <http://www.zabbix.com/download.php>。

下载后, 保存 http://www.zabbix.com/downloads/2.2.0/zabbix_agents_2.2.0.win.zip 到本地, 解压到 C:\Program Files\, 如图 3-29 所示。

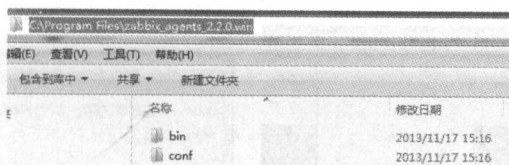


图 3-29

zabbix_agentd.conf 的配置和 Linux 中的配置一样, 此处不再重复, 步骤依然是先配置 zabbix_agentd.conf, 再启动服务。

2. 注册服务

进入 cmd 命令行 (见图 3-30)。

```
cmd> zabbix_agentd.exe --install -c "c:\Program Files\zabbix_agents_2.2.0.win\conf\zabbix_agentd.win.conf"
```

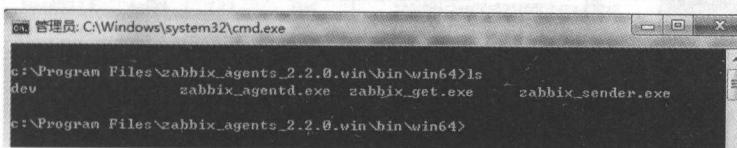


图 3-30

注册 Zabbix-Agent 服务, 注意, 路径中有空格, 应该用双引号 (见图 3-31)。

```
c:\Program Files\zabbix_agents_2.2.0.win\bin\win64>zabbix_agentd.exe --install -c "c:\Program Files\zabbix_agents_2.2.0.win\conf\zabbix_agentd.win.conf"
zabbix_agentd.exe [4896]: service [Zabbix Agent] installed successfully
zabbix_agentd.exe [4896]: event source [Zabbix Agent] installed successfully
c:\Program Files\zabbix_agents_2.2.0.win\bin\win64>
```

图 3-31

在服务管理界面中查看是否已经添加完成 (见图 3-32)。

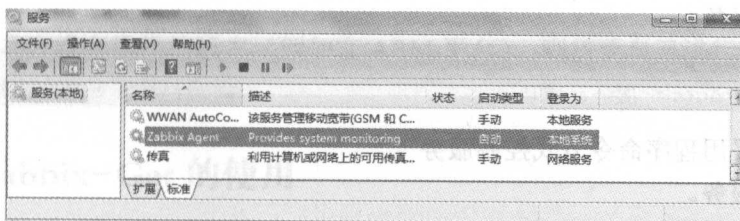


图 3-32

3. 启动服务

用 Windows 图形界面对服务进行控制(见图 3-33,也可以用命令行启动服务)。



图 3-33

(1) 采用 Windows 的 net 命令进行控制
启动服务。

```
C:\> net start "Zabbix Agent"
Zabbix Agent 服务正在启动 .
Zabbix Agent 服务已经启动成功。
```

停止服务。

```
C:\> net stop "Zabbix Agent"
Zabbix Agent 服务已成功停止。
```

(2) 采用程序命令方式控制服务
启动服务。

```
c:\Program Files\zabbix_agents_2.2.0.win\bin\win64> zabbix_agentd.
exe -s -c "c:\Program Files\zabbix_agents_2.2.0.win\conf\zabbix_agen
td.win.conf"
```

```
Zabbix_agentd.exe [8456]: service [Zabbix Agent] started successfully
```

停止服务。

```
c:\Program Files\zabbix_agents_2.2.0.win\bin\win64> zabbix_agentd.exe -x -c "c:\Program Files\zabbix_agents_2.2.0.win\conf\zabbix_agentd.win.conf"
Zabbix_agentd.exe [9040]: service [Zabbix Agent] stopped successfully
```

3.6 其他平台的安装

其他平台的安装方法也相同，到 <http://www.zabbix.com/download.php> 中下载对应的二进制版本安装并配置即可，如图 3-34 所示。

OS	Package	Release	Hardware	MD5	Download
AIX 6.1	Agents	2.2.1	powerpc	c8eb5995ff65140c9f2db41fcb835ffd	Download
FreeBSD 4.2	Agents	2.2.1	i386	a7735e33ed8c93727acfae52deab71d	Download
FreeBSD 6.2	Agents	2.2.1	i386	b26db2f16fdd7bc64b2d1dec17692c28	Download
FreeBSD 7.1	Agents	2.2.1	amd64	0908b23d26c3f70e7bb2a6e90fbd24f3	Download
FreeBSD 7.1	Agents	2.2.1	i386	b2dc484c3ee28a595755f3b40a450fa2	Download
FreeBSD 8.2	Agents	2.2.1	i386	56339a6263bafa74b3c6584aa18658d	Download
HP-UX 11.23	Agents	2.2.1	ia64	25a5d7b0a5f29c17275baedd458ffc91	Download
HP-UX 11.23	Agents	2.2.1	risc	3aa046d40ca9ef7473fe544b61adfbb5	Download
HP-UX 11.31	Agents	2.2.1	ia64	02a5455dcb4ed86ac9cd36a0a806c78	Download
HP-UX 11.31	Agents	2.2.1	risc	2c25e02ec31cdfd62b9435bcd886d24c	Download
Linux 2.4.x	Agents	2.2.1	i386	903889778bf6e7c9173e62f577e2c89	Download
Linux 2.6	Agents	2.2.1	amd64	96a7d73e575f818265ffd7a3d2abcb9f	Download
Linux 2.6	Agents	2.2.1	i386	e8623a4c203a0741b7b6bb9c91f604097	Download
Linux 2.6.23	Agents	2.2.1	amd64	9719c9ff1d9a6bad558d791eb65a3a24	Download
Linux 2.6.23	Agents	2.2.1	i386	909e888141734debfd0d95e4f19f53064	Download
NetBSD 5.0	Agents	2.2.1	i386	ee1db6dc388e88d97f1f0f32639805b	Download
OpenBSD 3.9	Agents	2.2.1	i386	821e3e38001a0223bcb2a6854108e4fa	Download
OpenBSD 4.3	Agents	2.2.1	i386	3a95711cec21a5266b4a78a926b71d70	Download
OpenBSD 4.7	Agents	2.2.1	amd64	ade4fb8af78b1b46083b1d4f5c2be60	Download
Solaris 9	Agents	2.2.1	sparc	f34efb5099c04b1de44170e56b187b44	Download
Solaris 10	Agents	2.2.1	amd64	e48499fcdcbf847aa438ad7cab5c9335	Download
Solaris 10	Agents	2.2.1	sparc	bb7a17d7e850ae4e9846df615fd70b8f	Download
Solaris 11	Agents	2.2.1	amd64	a7edcae0267a88d8016267e37536743d	Download
Windows (All)	Agents	2.2.1	i386,amd64	c32f48b57ca0ee671551c3589c9c9e6c	Download

图 3-34

对于官方没有提供的版本（例如在 ARM 平台），则需要自己进行编译安装，源码安装请读者参考第 16 章。

3.7 Zabbix-Get 的使用

Zabbix-Get 是 Zabbix 中的一个程序，用于 Zabbix-Server 到 Zabbix-Agent 的数据获取，通常可以用来检测验证 Agent 的配置是否正确。

用法如下。

```
zabbix_get [-hV] -s <host name or IP> [-p <port>] [-I <IP address>]
-k <key>
```

- s: 远程 Zabbix-Agent 的 IP 地址或者是主机名。
- p: 远程 Zabbix-Agent 的端口。
- I: 本机出去的 IP 地址, 用于一台机器中有多个网卡的情况。
- k: 获取远程 Zabbix-Agent 数据所使用的 Key。

示例如下。

```
shell# zabbix_get -s 192.168.0.240 -k system.uname
Linux zabbix.itnihao.com 2.6.32-358.el6.x86_64 #1 SMP Fri Feb 22 0
0:31:26 UTC 2013 X86_64
shell# zabbix_get -s 192.168.0.103 -k system.uname
Windows ITNIHAO-COM 6.1.7601 Microsoft Windows 7 Ultimate Edition
Service Pack 1 x64
shell# zabbix_get -s 192.168.0.240 -p 10050 -I 127.0.0.1 -k syste
m.uname
Linux zabbix.itnihao.com 2.6.32-358.el6.x86_64 #1 SMP Fri Feb 22 0
0:31:26 UTC 2013 X86_64
```

上面的源 IP 是 127.0.0.1 为访问本机, 如果是获取远程机器, 则 -I 后面的参数为外网网卡 IP。

用 `zabbix_get` 命令可以很方便地知道 key 是否能正常获取到数据, 这在测试自定义监控的时候特别有用。

3.8 Zabbix 相关术语 (命令)

1. zabbix_server

`zabbix_server` 是 Zabbix 服务端的核心程序。

2. zabbix_proxy

`zabbix_proxy` 是 Zabbix 代理服务的程序, 用于分布式监控 proxy 模式中。

3. zabbix_agent

`zabbix_agent` 是用超级服务 (xinetd) 的方式来启动的, 对应的配置文件为 `zabbix_agent.conf`。

`zabbix_agentd` 是以独立进程的方式来启动的, 对应的配置文件为 `zabbix_agentd.conf`。

`zabbix_agent` 是 Zabbix 专用客户端的程序。

4. zabbix_java_gateway

Zabbix 的 Java 采集服务端, 用于 JMX 的监控方式。

5. zabbix_sender

Zabbix 的 Trapping 模式,将采集到的数据通过定时任务等主动发送给 zabbix_server, 对于这种方式的使用, 请读者参考本书 6.3 节。

6. zabbix_get

zabbix_get 是一个数据获取测试命令, 相当于 snmp 中的 snmpwalk。

3.9 Zabbix-Server 对数据的存储

无论你采用什么架构, 是否使用代理, 数据存储对 Zabbix 来说, 都是一项非常大的挑战。通常来说, 使用分布式架构, 采用 Agent 端为主动模式, 会提高 Zabbix 的性能。关于构建分布式的架构和 Agent 端的主动模式, 请读者参考本书第 8 章。本节主要讲述如何对 Zabbix-Server 端数据的存储进行有效的优化 (即采用分表的方式)。

对于本节的内容, 推荐对 Zabbix 稍有经验的读者参考使用, 新手可以略过本节, 直接进入 3.10 节进行阅读。

3.9.1 Zabbix 对数据存储

通过前面的学习, 我们知道了 Zabbix-Server 将采集到的数据存储数据库中, 我们也了解到数据存储的大小与每秒处理的数据量有关, 所以数据存储取决于以下两个因素。

- Number of processed values per second (每秒处理的数据值): 更新数据。
- Housekeeper 的设置: 删除数据。

Zabbix-Server 将采集到的数据主要存储在 History 和 Trends 表中, 其表结构中的数据类型如表 3-7 所示。

表 3-7

表	数据类型	最大值范围	
history	保存历史数据	数字 (浮点数)	double(16,4)~999999999999.9999
history_uint	保存历史数据	数字 (非符号)	bigint(20) - 2 ⁶⁴ +1
history_str	保存短的字符串数据	字符	varchar(255)~255
history_text	保存长的字符串数据	文本	text~65535
history_log	保存日志字符串	日志	text~65535
trends	保存趋势数据	数字 (浮点数)	double(16,4)~999999999999.9999
trends_uint	保存趋势数据	数字 (非符号)	bigint(20) - 2 ⁶⁴ +1

另外，acknowledges、alerts、auditlog、events 和 service_alarms 表的数据也较大，故后文所提到的分表也会对这五个表进行区间划分。

在 History 表中，主要存储收集到的历史数据，而 Trends 主要存储经过计算的历史数据，如每小时数据的最小值、最大值和平均值。

History 的图像数据如图 3-35 所示，读取的是 History 表中的数据，图中的数据是每分钟（取决于数据采集的周期）的历史记录数据。

关于监控数据在前台页面的展示，读者可以通过分析 PHP 源码看到其实现过程（源码文件名是 include/graphs.inc.php）。

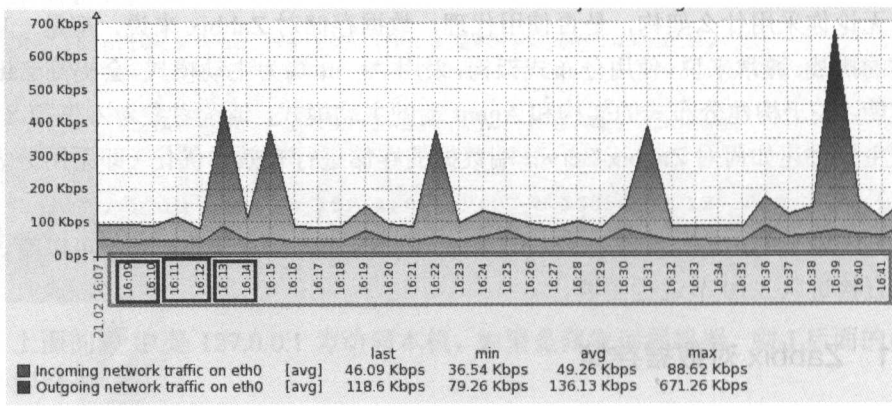


图 3-35

Trends 的图像数据如图 3-36 所示，读取的是 Trends 表中的数据，图中的数据是每三个小时的平均数据图。

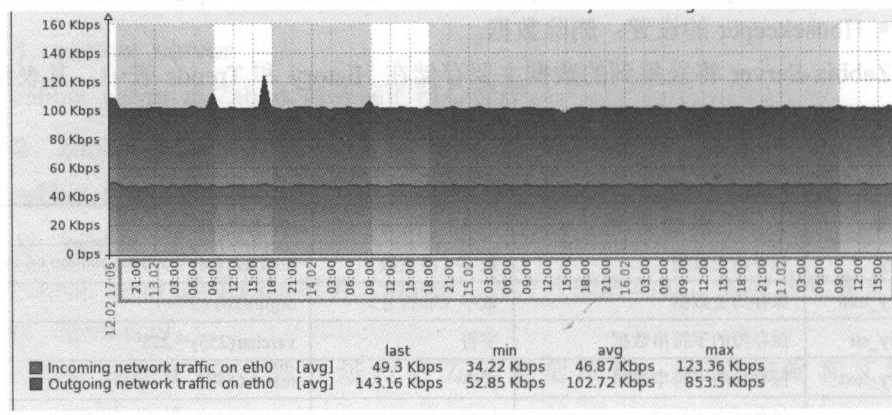


图 3-36

Trends 数据的存储有以下两个数据表。

- Trends（存储浮点数据类型）。

- trends_unit (存储非符号的整数)。

这两个表中都包含：最小值 (value_min)、最大值 (value_max) 和平均值 (value_avg)。

下面看看 Trends 表，其创建表的语句如下。

```
mysql> show create table trends\G;
***** 1. row *****
Table: trends
Create Table: CREATE TABLE `trends` (
  `itemid` bigint(20) unsigned NOT NULL,
  `clock` int(11) NOT NULL DEFAULT '0',
  `num` int(11) NOT NULL DEFAULT '0',
  `value_min` double(16,4) NOT NULL DEFAULT '0.0000',
  `value_avg` double(16,4) NOT NULL DEFAULT '0.0000',
  `value_max` double(16,4) NOT NULL DEFAULT '0.0000',
  PRIMARY KEY (`itemid`,`clock`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

同时，还有一个表 trends_uint，其创建表的语句如下。

```
mysql> show create table trends_uint\G;
***** 1. row *****
Table: trends_uint
Create Table: CREATE TABLE `trends_uint` (
  `itemid` bigint(20) unsigned NOT NULL,
  `clock` int(11) NOT NULL DEFAULT '0',
  `num` int(11) NOT NULL DEFAULT '0',
  `value_min` bigint(20) unsigned NOT NULL DEFAULT '0',
  `value_avg` bigint(20) unsigned NOT NULL DEFAULT '0',
  `value_max` bigint(20) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`itemid`,`clock`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

同样，History 也有以下几个表。

- history: 浮点数据。
- history_log: 日志。
- history_str: 字符串，255 个字符限制。
- history_text: 文本，不限制长度。
- history_uint: 无符号整数。

History 表结构如图 3-37 至图 3-39 所示。

```
mysql> desc history;
```

Field	Type	Null	Key	Default	Extra
itemid	bigint(20) unsigned	NO	MUL	NULL	
clock	int(11)	NO		0	
value	double(16,4)	NO		0.0000	
ns	int(11)	NO		0	

图 3-37

```
mysql> desc history_log;
```

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	MUL	NULL	
itemid	bigint(20) unsigned	NO	MUL	NULL	
clock	int(11)	NO		0	
timestamp	int(11)	NO		0	
source	varchar(64)	NO			
severity	int(11)	NO		0	
value	text	NO		NULL	
logeventid	int(11)	NO		0	
ns	int(11)	NO		0	

图 3-38

```
mysql> desc history_str;
```

Field	Type	Null	Key	Default	Extra
itemid	bigint(20) unsigned	NO	MUL	NULL	
clock	int(11)	NO		0	
value	varchar(255)	NO			
ns	int(11)	NO		0	

4 rows in set (0.00 sec)

```
mysql> desc history_text;
```

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	MUL	NULL	
itemid	bigint(20) unsigned	NO	MUL	NULL	
clock	int(11)	NO		0	
value	text	NO		NULL	
ns	int(11)	NO		0	

5 rows in set (0.00 sec)

```
mysql> desc history_uint;
```

Field	Type	Null	Key	Default	Extra
itemid	bigint(20) unsigned	NO	MUL	NULL	
clock	int(11)	NO		0	
value	bigint(20) unsigned	NO		0	
ns	int(11)	NO		0	

4 rows in set (0.00 sec)

图 3-39

在表 3-8 中，对 History 的常用字段进行了说明。

表 3-8

字段名称	数据类型	含 义
id	string	History 的 ID
clock	timestamp	接收到数据的时间
itemid	string	Items 的 ID
ns	integer	接收到数据的时间纳秒，Zabbix 2.0 以上版本支持，例如，日志，可以将时间精确到毫秒
value	string	接收到的数据

在本章的前面，我们已经学习了如何计算数据库的大小，基本上可以估算出 Zabbix-Server 的数据库会占用多大，但随着 Zabbix 数据库的增大，问题随之而来。

对于超过存储期限的数据，Zabbix-Server 用 Housekeeper 进程进行数据清理，读者可以分析源代码了解其实现过程（如图 3-40 所示），代码位置如下：

```
src/zabbix_server/housekeeper/housekeeper.c
```

```

static int housekeeping_history_and_trends(int now)
{
    const char * __function_name = "housekeeping_history_and_trends";
    int deleted = 0, i, rc;
    zbx_hk_history_rule_t *rule;
    zabbix_log(LOG_LEVEL_DEBUG, "in %s()", __function_name, now);
    /* prepare delete queues for all history housekeeping rules */
    hk_history_delete_queue_prepare_all(hk_history_rules; now);
    for (rule = hk_history_rules; rule != NULL; rule = rule->table; rule++)
    {
        if (ZBX_HK_OPTION_DISABLED == rule->option_mode)
            continue;
        /* process housekeeping rule */
        zbx_vector_ptr_sort(&rule->delete_queue, hk_item_update_cache_compare);
        for (i = 0; i < rule->delete_queue.values_num; i++)
        {
            zbx_hk_delete_queue_t *item_record = rule->delete_queue.values[i];
            rc = DBexecute("DELETE FROM zabbix.housekeeper WHERE item_id = %d AND clock = %d",
                rule->table, item_record->itemid, item_record->min_clock);
            if (ZBX_DB_OK == rc)
                deleted += rc;
        }
        /* clear history rule delete queue so it's ready for the next housekeeping cycle */
        hk_history_delete_queue_clear(rule);
    }
    zabbix_log(LOG_LEVEL_DEBUG, "in %s()", __function_name, deleted);
}

```

图 3-40

通过分析源代码，我们知道了 Zabbix-Server 对数据的清理主要是通过 DELETE 的 SQL 语句来执行删除动作。随着数据存储的越来越多，其执行效率会显著下降，有经验的读者都知道，在一个千万级、亿万级的表中执行一条 DELETE 的 SQL 语句，少则几十秒，多则几十分钟才能够完成，所以 Housekeeper 程序执行的 SQL 语句会严重影响 DB 的性能，从而导致数据库会成为整个监控系统的性能瓶颈。

对于很大的表，SQL 优化的方案中最常见的方式有横向扩展和纵向扩展，这两种方式中，一是用足够好的硬件，二是将数据进行分布式，而分表可以看作是分布式的一种，即按一定的规则将数据划分区间，从而避免全表扫描所带来的性能损失，最大程度地提高了性能。在这里采取的就是对表区间进行划分。

下面来看一个在线的 Zabbix 数据库中 History 表数据量的大小（见图 3-41）。

```

mysql> select table_name, (data_length+index_length)/1024/1024 as
total_mb, table_rows from information_schema.tables where table_
schema='zabbix';

```

history	110979.89062500	143414103
history_log	0.04687500	0
history_str	83.54687500	6411270
history_str_sync	0.04687500	0
history_sync	0.04687500	0
history_text	0.04687500	0
history_uint	20072.37500000	388921915
history_uint_sync	0.04687500	0
trends	285.00000000	4100497
trends_uint	756.00000000	10938711
trigger_depends	0.04687500	39
trigger_discovery	0.09375000	238
triggers	1.57812500	2297
user_history	0.03125000	3
users	0.03125000	4
users_groups	0.04687500	6
usrgrp	0.03125000	6
valuemaps	0.03125000	11

图 3-41

在 history_uint 表中，数据达到 3.8 亿条，如果在这个表中执行 DELETE 的 SQL 语句，其速度是可想而知的。

在了解对 Zabbix 的数据库进行分表的必要性之后，下面介绍如何划分表的区间。首先，对 Trends 表（见图 3-42）进行区间划分，这里分区标准是按天进行划分的（见图 3-43）。

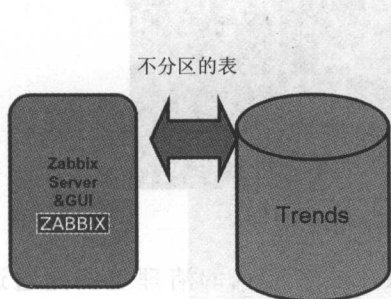


图 3-42

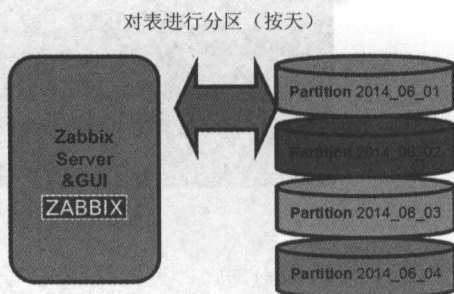


图 3-43

如果数据量不是特别大，也可以按月进行划分（见图 3-44）。

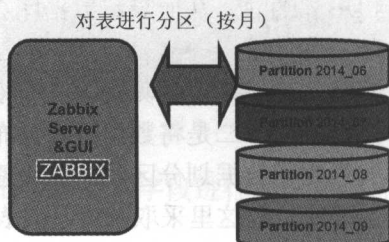


图 3-44

采用表分区后，需要关闭 Housekeeper 的功能，此时如果需要清理历史数据，只需要对表区间对应的历史期限数据进行删除即可。

注意，对于 Housekeeper 进程的关闭，可以在 Zabbix 2.0 的 zabbix_server.conf 中设置，设置完成后，重启 zabbix_server 服务，即可使修改后的配置生效。

```
### Option: DisableHousekeeping
# If set to 1, disables housekeeping.
#
# Mandatory: no
# Range: 0-1
DisableHousekeeping=1（值设置为1，关闭）
```

在 Zabbix 2.2 中，zabbix_server.conf 没有这个可配置的参数，Zabbix 2.2 的 housekeeper 是在 Web 界面中进行的配置（见图 3-45）。

在 Web 页面中，依次找到 Administration→General→Housekeeper，去掉勾选

状态,即可关闭 History 和 Trends 的 housekeeper 功能。

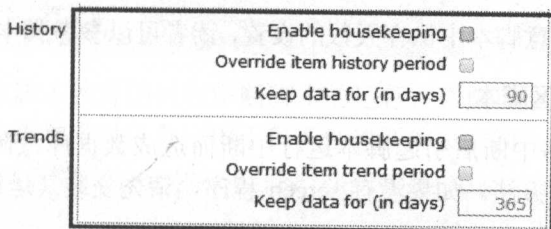


图 3-45

3.9.2 MySQL 表分区实例

下面用一个 MySQL 表分区脚本对 MySQL 进行表分区,该脚本来自以下地址。

http://blog.hbis.fr/2013/01/31/zabbix-mysql_partitioning/

1. 表分区脚本

脚本下载(参照 <https://github.com/itnihao/zabbix-book/blob/master/03-chapter/>)命令如下。

```
shell# wget https://raw2.github.com/cdand/zabbixdbpartitioning/master/partitiontables.sh
```

脚本具备的功能如下。

- ① 备份数据库。
- ② 对表进行分区间。
- ③ 添加定时任务。

注:该脚本在 Zabbix 2.0.6 和 Zabbix 2.2.2 中均测试通过,适合于已经安装过 Zabbix,但未分区的数据库;对于已经在线运行的环境,Zabbix 数据库中的表数据量会较大,执行此脚本时间会非常长,笔者测试过的有 10 多个小时还在执行,故建议读者先清空表数据(注意历史记录会全部被清空,或者备份数据库),再执行,清空语句如下。

```
mysql> use zabbix;
mysql> truncate table history;
mysql> optimize table history;
mysql> truncate table history_str;
mysql> optimize table history_str;
mysql> truncate table history_uint;
mysql> optimize table history_uint;
mysql> truncate table trends;
mysql> optimize table trends;
mysql> truncate table trends_uint;
mysql> optimize table trends_uint;
```



```
mysql> truncate table events;
mysql> optimize table events;
```

另外，需要注意脚本中保存天数的设置，读者可以参考脚本的注释自行设置。

2. 运行表分区脚本

为了防止网络中断后引起脚本运行中断而造成数据库故障，我们应该选用 screen 后台执行的方法。如果没有 screen 程序，请先安装（运维人员要处处持有谨慎的态度）。

```
shell# screen -R zabbix
shell# bash partitiontables.sh
```

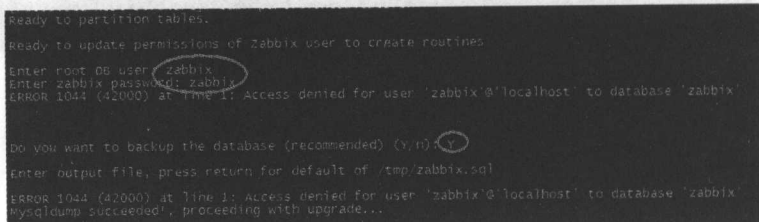
退出 screen，脚本将在后台执行，方法如下：

按组合键 CTRL+A 之后再按组合键 CTRL+D

进入 screen，可以查看后台运行的任务：

```
shell# screen -R zabbix
```

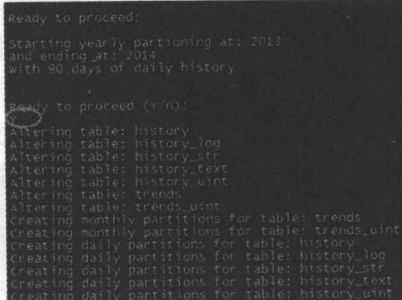
输入用户名为 zabbix，密码为 zabbix，选择备份数据库，连续按两次回车键（见图 3-46 和图 3-47），观察 /tmp/zabbix.sql 会逐渐增大。



```
Ready to partition tables.
Ready to update permissions of Zabbix user to create routines
Enter root DB user: zabbix
Enter zabbix password: zabbix
ERROR 1043 (42000) at line 1: Access denied for user 'zabbix'@'localhost' to database 'zabbix'

Do you want to backup the database (recommended) (y/n): y
Enter output file, press return for default of /tmp/zabbix.sql
ERROR 1043 (42000) at line 1: Access denied for user 'zabbix'@'localhost' to database 'zabbix'
mysqldump succeeded!, proceeding with upgrade...
```

图 3-46



```
Ready to proceed (y/n): y
Starting yearly partitioning at: 2013
and ending at: 2014
with 90 days of daily history

Ready to proceed (y/n): y
Altering table: history
Altering table: history_log
Altering table: history_str
Altering table: history_text
Altering table: history_uint
Altering table: trends
Altering table: trends_uint
Creating monthly partitions for table: trends_uint
Creating daily partitions for table: history
Creating daily partitions for table: history_log
Creating daily partitions for table: history_str
Creating daily partitions for table: history_text
Creating daily partitions for table: history_uint
```

图 3-47

当脚本中用 zabbix 用户去连接数据库时，会出现访问被拒绝的提示，主要是因为以下两条 SQL 语句无法执行，解决方法是可以忽略，直接按回车键即可。

```
mysql -B -h $DBHOST -u $DBADMINUSER -p$DBADMINPASS -e "GRANT CREATE ROUTINE ON zabbix.* TO '$DBUSER'@'localhost';"
mysql -h $DBHOST -e "GRANT LOCK TABLES ON zabbix.* TO '$DBUSER'@'$DBHOST' IDENTIFIED BY '$DBPASS';"
```

注意：严禁在脚本执行的过程中中断脚本的运行，否则可能造成表的损坏。脚本运行完毕后，会添加定时任务（用于每天创建新的表空间）。

```
[root@localhost ~]# cat /etc/cron.daily/zabbixhousekeeping
#!/bin/bash
/usr/local/zabbix/cron.d/housekeeping.sh
[root@localhost ~]# cat /usr/local/zabbix/cron.d/housekeeping.sh
#!/bin/bash

MAILTO=Y
tmpfile=/tmp/housekeeping$$

date >$tmpfile
/usr/bin/mysql --skip-column-names -B -h localhost -u zabbix -pzabbix zabbix -e "CALL create_zabbix_partitions();" >>$tmpfile 2>&1
/usr/bin/mail -s "Zabbix MySQL Partition Housekeeping" $MAILTO <$tmpfile
rm -f $tmpfile
```

为了验证表分区的 SQL 触发器能否正常运行（如果你的 MySQL 是自己编译安装，由于 sock 文件等位置为非标准，会导致此 SQL 语句执行失败，需要修改此脚本，可增加--socket-路径），通过手动方式运行命令调用触发器，验证表分区能否正常创建，命令如下。

```
shell# mysql --skip-column-names -B -h localhost -u zabbix -pzabbix zabbix -e "CALL create_zabbix_partitions();" #如图3-48所示
```

```
[root@localhost software]# /usr/bin/mysql --skip-column-names -B -h localhost -
create_partition(zabbix,history,p20131215,1387123200)
create_partition(zabbix,history,p20131216,1387209600)
create_partition(zabbix,history,p20131217,1387296000)
create_partition(zabbix,history,p20131218,1387382400)
create_partition(zabbix,history,p20131219,1387468800)
create_partition(zabbix,history_log,p20131215,1387123200)
create_partition(zabbix,history_log,p20131216,1387209600)
create_partition(zabbix,history_log,p20131217,1387296000)
create_partition(zabbix,history_log,p20131218,1387382400)
create_partition(zabbix,history_log,p20131219,1387468800)
create_partition(zabbix,history_str,p20131215,1387123200)
create_partition(zabbix,history_str,p20131216,1387209600)
create_partition(zabbix,history_str,p20131217,1387296000)
create_partition(zabbix,history_str,p20131218,1387382400)
create_partition(zabbix,history_str,p20131219,1387468800)
create_partition(zabbix,history_text,p20131215,1387123200)
create_partition(zabbix,history_text,p20131216,1387209600)
create_partition(zabbix,history_text,p20131217,1387296000)
create_partition(zabbix,history_text,p20131218,1387382400)
create_partition(zabbix,history_text,p20131219,1387468800)
create_partition(zabbix,history_uint,p20131215,1387123200)
create_partition(zabbix,history_uint,p20131216,1387209600)
create_partition(zabbix,history_uint,p20131217,1387296000)
create_partition(zabbix,history_uint,p20131218,1387382400)
create_partition(zabbix,history_uint,p20131219,1387468800)
create_partition(zabbix,trends,p201403,1396281600)
create_partition(zabbix,trends_uint,p201403,1396281600)
```

图 3-48


```
| count(*) |
+-----+
| 6302 |
+-----+
1 row in set (0.13 sec)
```

至此，表分区已经完成，对提高数据库的性能具有重要的作用。对 MySQL 的优化还可以通过调整本身的配置参数来进行。

若读者想了解更多关于表分区知识，可以参考以下链接。

https://www.zabbix.org/wiki/Docs/howto/mysql_partition

3.10 Zabbix init 脚本解释

在 Zabbix 源码包中可以看到启动脚本，如图 3-50 所示。

```
itnihao@itnihao zabbix-2.2.4$ tree misc/init.d/
misc/init.d/
├── README
├── aix
│   └── zabbix_agentd
├── debian
│   ├── zabbix-agent
│   └── zabbix-server
├── fedora
│   ├── core
│   │   ├── zabbix_agentd
│   │   └── zabbix_server
│   ├── core5
│   │   ├── zabbix_agentd
│   │   └── zabbix_server
│   └── RHEL5.CentOS5用此脚本
├── freebsd
│   ├── zabbix_agentd
│   └── zabbix_server
├── gentoo
│   ├── zabbix-agentd
│   └── zabbix-server
├── suse
│   ├── 9.1
│   │   ├── zabbix_agentd
│   │   └── zabbix_server
│   ├── 9.2
│   │   ├── zabbix_agentd
│   │   └── zabbix_server
│   └── 9.3
│       ├── zabbix_agentd
│       └── zabbix_server
├── tru64
│   ├── zabbix_agentd
│   └── zabbix_server
├── ubuntu
│   ├── zabbix-agent.conf
│   └── zabbix-server.conf
```

图 3-50

如果是源码安装，需要将脚本复制到/etc/init.d/目录（类 RedHat 系统）下。

如果路径不同，修改 BASEDIR=/usr/local 为实际路径。如 Zabbix 安装在 /opt/monitor/zabbix 目录下，则修改为 BASEDIR=/opt/monitor/zabbix 即可。

服务脚本中修改的参数如下。

```
# Variables
# Edit these to match your system settings
# Zabbix-Directory
BASEDIR=/usr #将此参数修改为实际路径
```

```
# Binary File
BINARY_NAME=zabbix_server

# Full Binary File Call
FULLPATH=$BASEDIR/sbin/$BINARY_NAME

# PID file
PIDFILE=/var/run/zabbix/$BINARY_NAME.pid
```

3.11 高可用和安全

1. 高可用

由于 Zabbix 运行在单节点上，难免会造成服务的不可用，对于要求比较严格的生产环境，配置服务的高可用成为必须的要求。对于这个问题，将在 16.3.3 节中讨论。

2. 安全

由于 Zabbix-Server、Zabbix-Proxy、Zabbix-Agent 之间的通信会涉及跨网络、跨地域的大型分布式环境，会面临互联网的复杂环境，所以各服务之间的通信安全成了必须考虑的因素。

为了使 Zabbix 之间的通信采用安全方式，可以考虑以下方法。

- Zabbix-Server 和 Zabbix-Proxy 之间通过 VPN 连接。
- Zabbix-Server 和 Zabbix-Proxy 之间通过 SSH 通道连接。
- Zabbix-Server 和 Zabbix-Proxy 之间通过 Stunnel 连接。
- 设置严格的防火墙策略，只允许特定的服务器访问，如 Zabbix-Server 只允许 Zabbix-Proxy 访问或者只允许特定的网段访问。
- 时常关注 Zabbix 官方的漏洞信息，及时更新补丁。

关于 stunnel 的方式，可参考以下网址。

<http://www.mrvoip.com.au/blog/secure-zabbix-proxy-communications-stunnel>

3. 禁用 Zabbix 的重新安装

```
shell# vim /usr/share/zabbix/include/menu.inc.php
#注销如图3-51所示的内容
```

```
/*
 * label' => _('Installation')
 */
array(
    'url' => 'setup.php',
    'label' => _('Installation')
)*/
```

图 3-51

可以看到，菜单上已经没有安装按钮了，如图 3-52 所示。

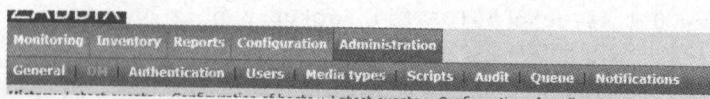


图 3-52

3.12 Zabbix 数据库的备份

备份 Zabbix 数据库可以防止数据库因意外损坏而导致所有的配置数据丢失的严重后果，备份目的是用于灾难恢复。这里选择对单表备份，而非全部备份。排除以下的表（原因是以下表为监控的历史展示数据，非配置类数据）。

```
history* trends* Acknowledges Alerts Auditlog Events service_alarms
```

备份脚本如下（读者可以根据需要进行修改）。

```
#!/bin/bash
#author: itnihao
#mail: itnihao@qq.com
#http://www.itnihao.com
#https://github.com/itnihao/zabbix-book/blob/master/03-chapter/Z
abbix_MySQLdump_per_table.sh

source /etc/bashrc && source /etc/profile

MySQL_USER=zabbix
MySQL_PASSWORD=zabbix
MySQL_HOST=localhost
MySQL_PORT=3306
MySQL_DUMP_PATH=/mysql_backup
MySQL_DATABASE_NAME=zabbix
DATE=$(date '+%Y-%m-%d')

[ -d ${MySQL_DUMP_PATH} ] || mkdir ${MySQL_DUMP_PATH}
cd ${MySQL_DUMP_PATH}
[ -d logs ] || mkdir logs
[ -d ${DATE} ] || mkdir ${DATE}
cd ${DATE}

TABLE_NAME_ALL=$(mysql -u${MySQL_USER} -p${MySQL_PASSWORD} -P${MySQL_PORT} -h${MySQL_HOST} ${MySQL_DATABASE_NAME} -e "show tables"|egrep -v "(Tables_in_zabbix|history*|trends*|acknowledges|alerts|auditlog|events|service_alarms)")
for TABLE_NAME in ${TABLE_NAME_ALL}
do
    mysqldump -u${MySQL_USER} -p${MySQL_PASSWORD} -P${MySQL_PORT} -h${MySQL_HOST} ${MySQL_DATABASE_NAME} ${TABLE_NAME} >${TABLE_NAME}.sql
    sleep 1
```



```
done

[ "$?" == 0 ] && echo "${DATE}: Backup zabbix succeed" >> ${MySQL_DUMP_PATH}/logs/ZabbixMysqldump.log
[ "$?" != 0 ] && echo "${DATE}: Backup zabbix not succeed" >> ${MySQL_DUMP_PATH}/logs/ZabbixMysqldump.log

cd ${MySQL_DUMP_PATH}/
rm -rf $(date +%Y%m%d --date='5 days ago')
exit 0
```

第 4 章 快速配置和使用

本章在第 3 章的基础上,介绍 Zabbix 的快速配置使用,让读者了解配置流程,通过本地浏览器访问 <http://ServerIP/zabbix> 开始配置和使用 Zabbix。

默认的用户名为 Admin,密码是 zabbix,其用户名和密码存于数据库中。

```
mysql> SELECT * FROM zabbix.users WHERE alias='Admin';
```

如果忘记 Admin 用户名和密码,可以直接修改数据库密码字段,命令如下:

```
mysql> UPDATE zabbix.users SET passwd=md5('zabbix') WHERE alias='Admin';  
mysql> FLUSH PRIVILEGES;
```

使用 Zabbix 进行监控之前,要理解 Zabbix 监控的流程。

4.1 配置流程

Zabbix 完整的监控配置流程可以简单描述为:

Host groups (主机组) → Hosts (主机) → Applications (监控项组) → Items (监控项) → Triggers (触发器) → Event (事件) → Actions (处理动作) → User groups (用户组) → Users (用户) → Medias (告警方式) → Audit (日志审计)。

在实际使用的时候,Items、Trigger、Graph 通常采用模板进行监控配置,模板的特点就是可以对相同需求的监控项重复使用,无须对每台主机进行逐个设置。

对于使用 Zabbix 来说,配置 Graph 不是必需的,因为没有配置图形,数据获取也不影响,数据获取是 Items 的功能,但是对用户 (Zabbix 的 Web 界面用户) 来说,没有图形,就无法看到可视化的数据,因此,需要对最关心的 Items 添加图形,以便将数据可视化。

配置 Trigger 不是必需的,但是对于特别关注的数据库,需要对取到的值进行条件判断,这时配置触发器就是必需的。

另外,如果想集中查看图形,可以使用 Screens 功能 (遗憾的是, Screens 功能并不十分完美,只能满足一般需求)。

下面用一张图来展示 Zabbix 的运行流程,如图 4-1 所示,该图更能体现各种逻辑关系。

读者当前只需要对这个流程图有一个简单的印象即可，待学习完本章，再回过头来看这个图，会加深理解。

Zabbix配置流程图

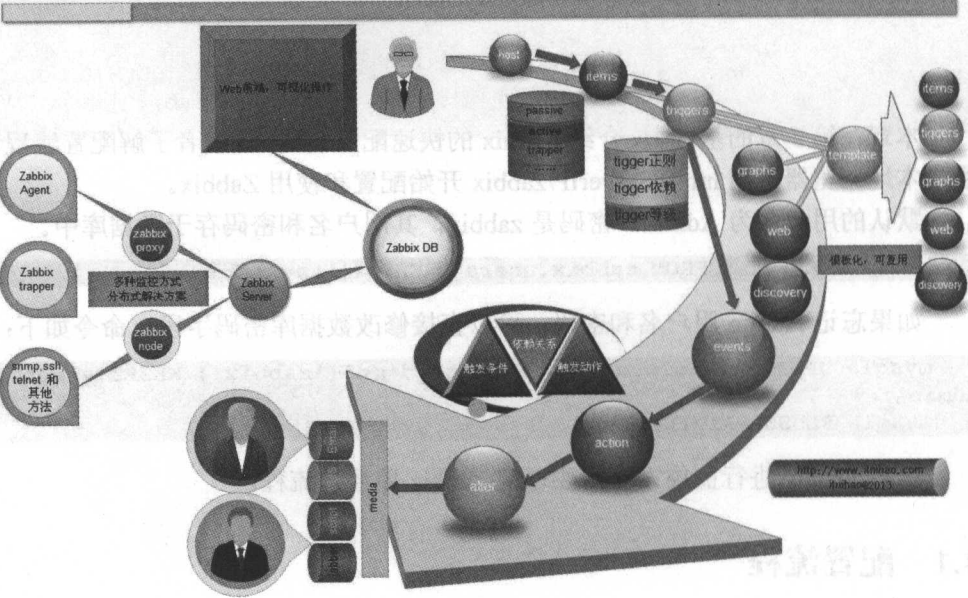


图 4-1

简单的逻辑可以用图 4-2 表示。

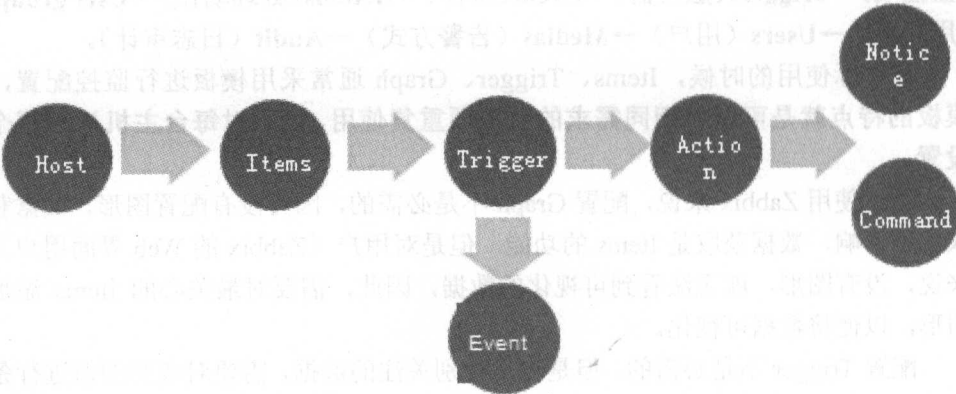


图 4-2

下面我们来配置 Zabbix。

4.2 主机组的添加

分组的目的是将同一属性的主机归类，如果你不想将新添加的主机归纳为自定义的分组，可以使用默认的分组。

配置主机组，打开前端 Web 页面，单击 Configuration→Host groups，如图 4-3 所示。

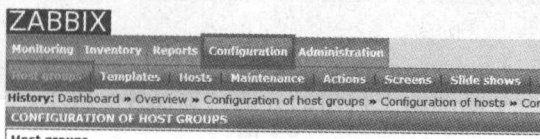


图 4-3

主机组中可以包含主机，也可以包含模板。

建议：对同一属性的主机或者模板，尽量归纳到分组，方便以后管理。分组原则如下。

- 以地理位置的纬度进行划分。
- 以业务为单位划分组。
- 以机器用途划分。
- 以系统版本划分。
- 以应用程序划分组。
- 其他方式等。

按组划分的好处是方便分组管理，如图 4-4 所示，就是以组划分的实例。

Monitoring Inventory Reports Configuration Administration		
Host groups	Templates	Hosts Maintenance Actions Screens Slide shows Maps
History: Dashboard » Overview » Configuration of host groups » Configuration of hosts » Configuration of host groups		
CONFIGURATION OF HOST GROUPS		
Host groups		
Displaying 1 to 9 of 9 found		
Name	#	Members
Discovered hosts	Templates (0) Hosts (1)	devops.itnhaio.com
Hypervisors	Templates (0) Hosts (0)	
Linux Cluster App	Templates (1) Hosts (0)	Template App HAProxy v1.1
Linux servers	Templates (0) Hosts (0)	
Proxy	Templates (0) Hosts (0)	
Templates	Templates (39) Hosts (1)	Template OS Linux, Template App Zabbix Server, Template App Zabbix SNMP Device, Template SNMP OS Windows, Template SNMP Disks, Template SR1630, Template App MySQL, Template OS OpenBSD, Template OS OS Windows, Template JMX Generic, Template JMX Tomcat, Template Template App HTTP Service, Template App HTTPS Service, Template Template App POP Service, Template App SMTP Service, Template A
		devops.itnhaio.com

图 4-4

如何添加新主机组呢？方法是单击 Host groups 上的按钮 Create host group，如图 4-5 所示。

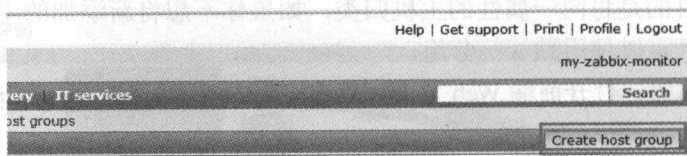


图 4-5

单击后会出现一个添加新主机组的界面，如图 4-6 所示。

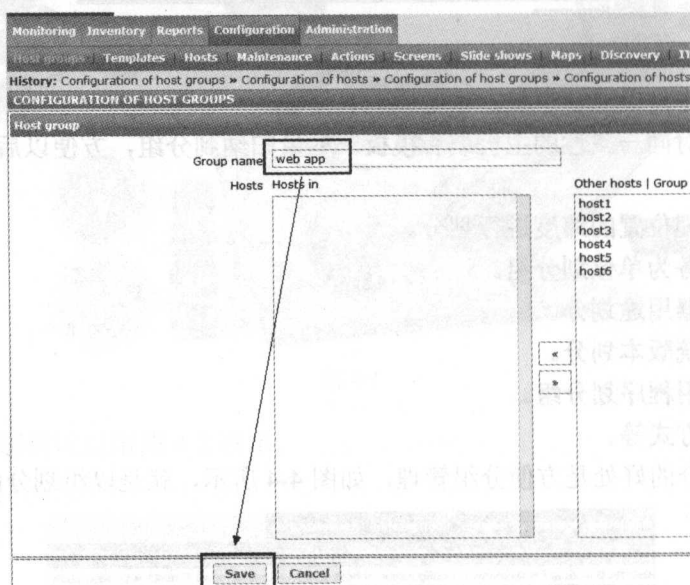


图 4-6

在图 4-6 中，“Group name”是组的名称；“Hosts Hosts in”是选择哪些设备属于这个新添加的组。

单击 Save 保存，新添加的主机组就可以显示在主机组列表中，如图 4-7 所示。

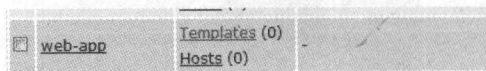


图 4-7

当单击主机组中的信息名称时，会跳转到相关的页面。例如，单击 Hosts，会跳转到属于此组的 Host 主机列表。

如果想在数据库中查看主机组，运行下面这条语句即可。

```
mysql> SELECT * FROM zabbix.groups; #如图4-8所示
```

```
mysql> select * from zabbix.groups;
```

groupid	name	internal	flags
1	Templates	0	0
2	Linux servers	0	0
4	Zabbix servers	0	0
5	Discovered hosts	1	0
6	Virtual machines	0	0
7	Hypervisors	0	0
8	business manager group1	0	0
9	business manager group2	0	0
10	business manager group3	0	0
11	DevOps	0	0
12	Dev web app group1	0	0
13	Dev web app group2	0	0
14	Dev web app group3	0	0
15	web app	0	0

图 4-8

4.3 模板的添加

监控项、触发器、图形、Web、Discovery 等都是存在于主机之上的，由于多个主机都会用相同的监控配置，因此，可以对这部分同类的数据进行归纳抽象，即将这些数据做成模板。当我们需要对其他监控数据进行配置的时候，只需要对相应的主机添加对应的模板即可。

配置模板的步骤为：单击 Configuration→Templates→Create template，如图 4-9 所示。

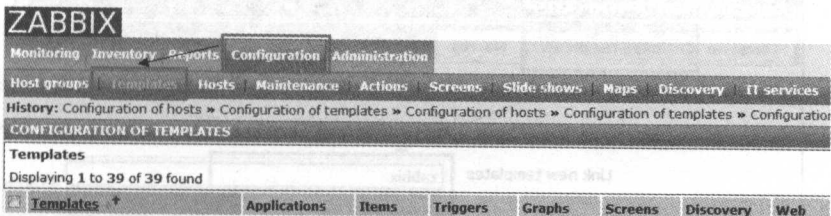


图 4-9

创建模板，依次单击 Templates→Create template，如图 4-10 所示。

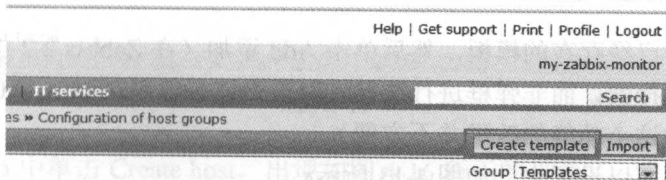


图 4-10

填入模板名称和所属的组，如图 4-11 所示。

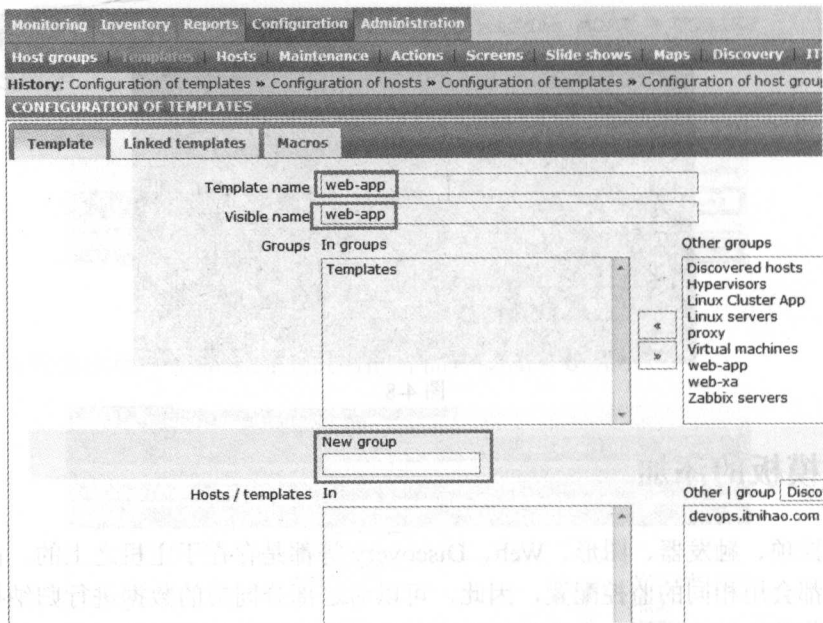


图 4-11

模板具有继承的功能（将一个模板在另外一个模板中使用），如图 4-12 所示。

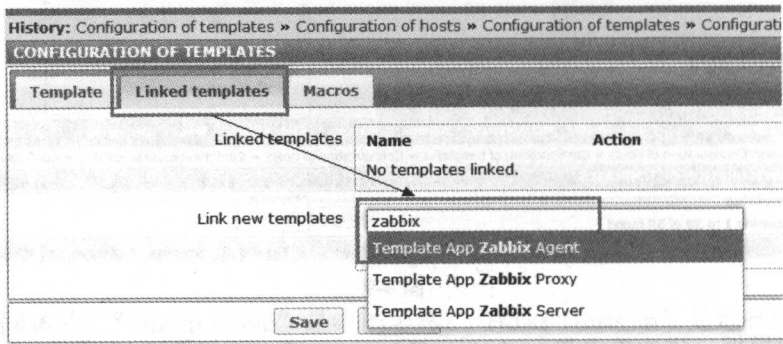


图 4-12

搜索当前已经存在的模板，然后单击 Add 添加（在 Zabbix 2.2 中，将添加数据的方式改为搜索，而非弹框进行选择，对于习惯了使用 Zabbix 2.0 之前版本的读者，会感觉这个功能的改进并不方便）。

模板中也可以设置宏，如图 4-13 所示。

宏主要是对变量的定义，设置宏的作用是方便后面在 Items、Trigger 中引用，在模板中配置一个宏，在不同的主机对该宏设置不同的变量值，从而达到模板化通用的目的。

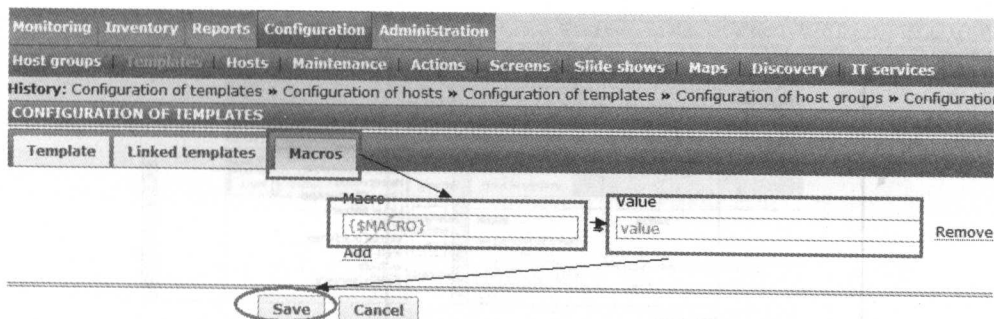


图 4-13

注意，宏的名称为{\$名称}，宏的字符范围为“A~Z、0~9、_、.”，如果不是这个范围的字符，一律是无效的（请读者参考 5.4 节的内容）。

查看新建的模板，如图 4-14 所示。可以看到，Items 中已经有三个 Items 了，是继承于 Zabbix Agent 模板的。关于如何在模板中添加其他的 Items，第 5 章将有更深入的讲解。

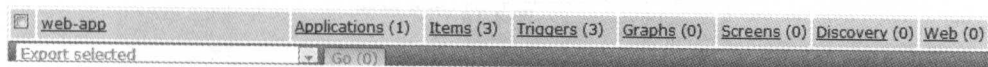


图 4-14

4.4 添加主机

添加主机的目的是对具体的设备进行监控，步骤为：依次单击 Configuration→Hosts→Create host，如图 4-15 所示。

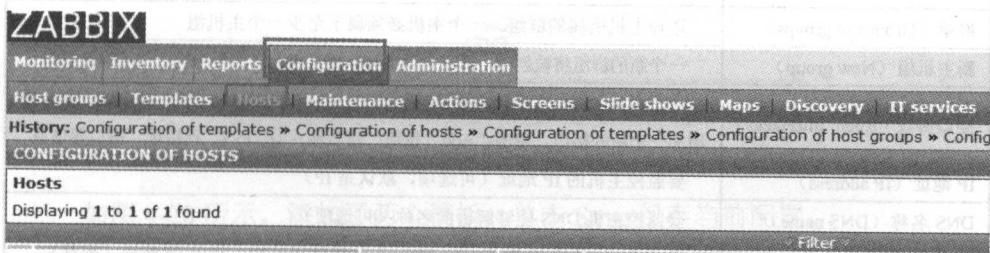


图 4-15

在图 4-15 中单击 Create host，出现如图 4-16 所示的界面。

图 4-16 中各参数的详细说明如表 4-1 所示。

The screenshot shows the 'CONFIGURATION OF HOSTS' window in Zabbix. The 'Host' tab is active. The 'Host name' field is empty. The 'Visible name' field is empty. The 'Groups' section has 'In groups' and 'Other groups' lists. The 'Agent interfaces' section has fields for 'IP address' (127.0.0.1), 'DNS name', 'Connect to' (IP), 'Port' (10050), and 'Default'. There are 'Add' buttons for 'Agent interfaces', 'SNMP interfaces', 'JMX interfaces', and 'IPMI interfaces'. The 'Monitored by proxy' dropdown is set to '(no proxy)'. The 'Status' dropdown is set to 'Monitored'. 'Save' and 'Cancel' buttons are at the bottom.

图 4-16

表 4-1

参 数	描 述		
主机名 (Host name)	输入一个不重复的主机名。只允许大小写字母、数字、标点符号和下划线。 注意：编辑该名称对应客户端的配置文件时，主机名 (Hostname) 这一项必须与此处输入的值相同。在主机存活检查时需要这个名字		
访问名 (Visible name)	如果设置该名字，那么它将出现在主机列表、地图等地方。该属性需要 UTF-8 支持		
群组 (Groups in groups)	选择主机所属的群组。一个主机必须属于至少一个主机组		
新主机组 (New group)	一个新的群组将被创建，然后自动链接到该主机上。如果为空，该项将被忽略		
接口协议 (Agent interfaces)	一个主机支持的主机接口协议类型包括：Agent、SNMP、JMX 和 IPMI，如果想增加一个新的接口，单击“Add”按钮，然后输入 IP/DNS、连接项、端口等信息		
IP 地址 (IP address)	要监控主机的 IP 地址（可选项，默认是 IP）		
DNS 名称 (DNS name)	要监控主机 DNS 能够解析的名称（可选项）		
与 Agent 通信的方式 (Connect to)	单击对应名称的按钮 (IP 或 DNS)	IP	连接要监控主机的 IP 地址（推荐）
		DNS	要监控主机能够正常解析的 DNS 名称
端口 (Port)	TCP 协议的端口，Zabbix 客户端使用的默认值是 10050		
通过代理服务器进行监控 (Monitored by proxy)	主机可以通过 Zabbix 服务器或者 Zabbix 的一个代理去监控客户端		
状态 (Status)	Monitored	主机是活动的，监控就绪	
	Not monitored	主机已停止，因此没被监控	

选择模板，如图 4-17 所示。

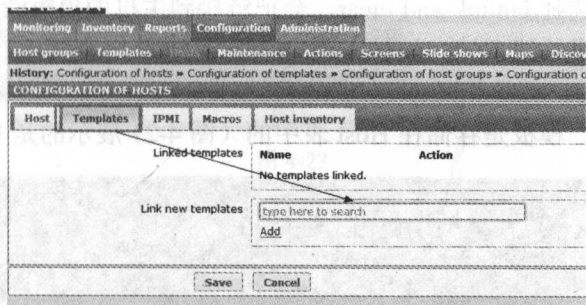


图 4-17

搜索模板名称，如图 4-18 所示。

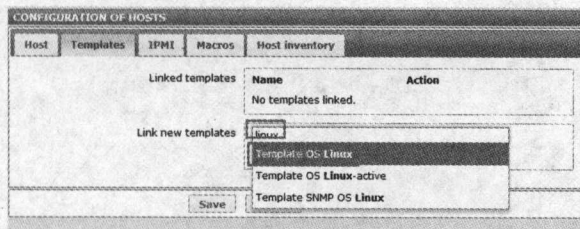


图 4-18

搜索出来后，单击 Add 添加该模板，如图 4-19 所示。

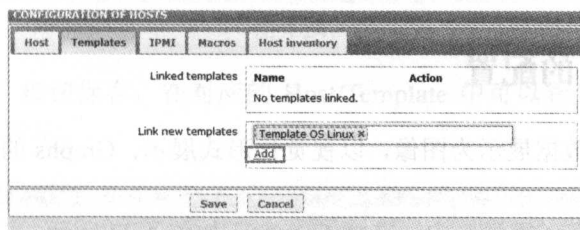


图 4-19

如图 4-20 所示，添加了两个模板，单击 Save 保存当前配置。

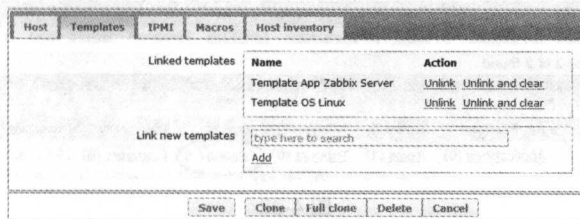


图 4-20

注意：如果要把模板从主机中删除，选择 Unlink，并不会把已经添加的监控数据给清理掉。选择 Unlink and clear，会把链接到主机的模板清理掉，并且会把主机中的监控数据给清理掉。如果还想选择配置 IPMI、Macros、Host inventory，依次选择配置即可，方法相同。

在数据库中，模板是存储在 Host 表中的（图 4-21 展示的是 Hosts 表结构）。

```

Hosts | CREATE TABLE `hosts` (
  `hostid` bigint(20) unsigned NOT NULL,
  `proxy_hostid` bigint(20) unsigned DEFAULT NULL,
  `host` varchar(64) NOT NULL DEFAULT '',
  `status` int(11) NOT NULL DEFAULT '0',
  `disable_until` int(11) NOT NULL DEFAULT '0',
  `error` varchar(128) NOT NULL DEFAULT '',
  `available` int(11) NOT NULL DEFAULT '0',
  `errors_from` int(11) NOT NULL DEFAULT '0',
  `lastaccess` int(11) NOT NULL DEFAULT '0',
  `ipmi_auth_type` int(11) NOT NULL DEFAULT '0',
  `ipmi_privilege` int(11) NOT NULL DEFAULT '0',
  `ipmi_username` varchar(16) NOT NULL DEFAULT '',
  `ipmi_password` varchar(20) NOT NULL DEFAULT '',
  `ipmi_disable_until` int(11) NOT NULL DEFAULT '0',
  `ipmi_available` int(11) NOT NULL DEFAULT '0',
  `snmp_disable_until` int(11) NOT NULL DEFAULT '0',
  `snmp_available` int(11) NOT NULL DEFAULT '0',
  `maintenanceid` bigint(20) unsigned DEFAULT NULL,
  `maintenance_status` int(11) NOT NULL DEFAULT '0',
  `maintenance_type` int(11) NOT NULL DEFAULT '0',
  `maintenance_from` int(11) NOT NULL DEFAULT '0',
  `ipmi_errors_from` int(11) NOT NULL DEFAULT '0',
  `snmp_errors_from` int(11) NOT NULL DEFAULT '0',
  `ipmi_error` varchar(128) NOT NULL DEFAULT '',
  `snmp_error` varchar(128) NOT NULL DEFAULT '',
  `jmx_disable_until` int(11) NOT NULL DEFAULT '0',
  `jmx_available` int(11) NOT NULL DEFAULT '0',
  `jmx_errors_from` int(11) NOT NULL DEFAULT '0',
  `jmx_error` varchar(128) NOT NULL DEFAULT '',
  `name` varchar(64) NOT NULL DEFAULT '',
  `flags` int(11) NOT NULL DEFAULT '0',
  `templateid` bigint(20) unsigned DEFAULT NULL,
  PRIMARY KEY (`hostid`),
  KEY `hosts_1` (`hostid`),
  KEY `hosts_2` (`status`),
  KEY `hosts_3` (`proxy_hostid`),
  KEY `hosts_4` (`name`),
  KEY `hosts_5` (`maintenanceid`),
  KEY `c_hosts_3` (`templateid`),
  CONSTRAINT `c_hosts_1` FOREIGN KEY (`proxy_hostid`) REFERENCES `hosts` (`hostid`),
  CONSTRAINT `c_hosts_2` FOREIGN KEY (`maintenanceid`) REFERENCES `maintenances` (`maintenanceid`),
  CONSTRAINT `c_hosts_3` FOREIGN KEY (`templateid`) REFERENCES `hosts` (`hostid`) ON DELETE CASCADE,
  ENGINE=InnoDB DEFAULT CHARSET=utf8
)
    
```

图 4-21

4.5 Graphs 的配置

Graphs 是将数据展示为图像，以视觉化形式展示，Graphs 的配置存在于主机和模板中。

在所在的主机（模板）中，选择 Graphs，如图 4-22 所示。

Monitoring Inventory Reports Configuration Administration						
Host groups Templates Hosts Maintenance Web Actions Screens Slide shows Maps D						
History: Dashboard » Configuration of host groups » Configuration of hosts » Configuration of templates » Conf						
CONFIGURATION OF HOSTS						
Hosts						
Displaying 1 to 2 of 2 found						
Filter						
<input type="checkbox"/>	Name	Applications	Items	Triggers	Graphs	Discovery Interface
<input checked="" type="checkbox"/>	trapper	Applications (0)	Items (1)	Triggers (0)	Graphs (0)	Discovery (0) 192.168.0.200: 10050

图 4-22

单击“Create graph”按钮，如图 4-23 所示。

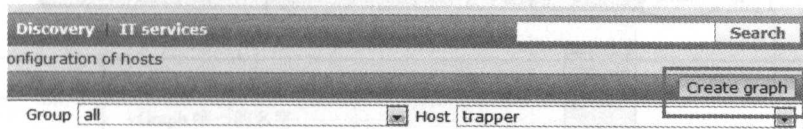


图 4-23

出现新建图形的界面，如图 4-24 所示，输入 Name，选择所需要添加的 Items。

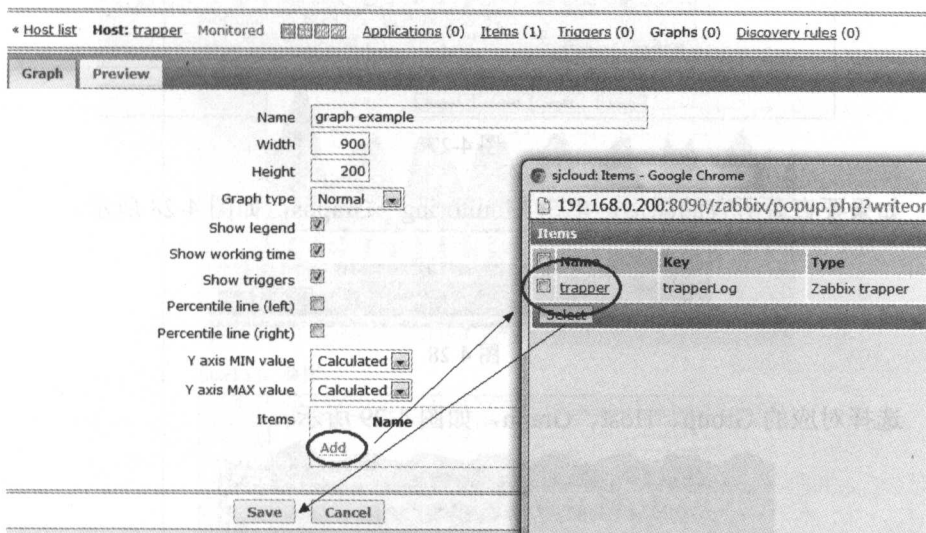


图 4-24

单击“Save”按钮保存，在对应的 Host/Template 中可以看到 Graphs 新增了内容，如图 4-25 所示。

Name	Applications	Items	Triggers	Graphs	Discovery	Interface
trapper	Applications (0)	Items (1)	Triggers (0)	Graphs (1)	Discovery (0)	192.168

图 4-25

单击 Graphs，可以看到其中已经新添加了一个 Graphs，如图 4-26 所示。

Name	Width
graph example	900

图 4-26

单击刚才创建的 Graphs，选择 Preview，即可查看图形，如图 4-27 所示。

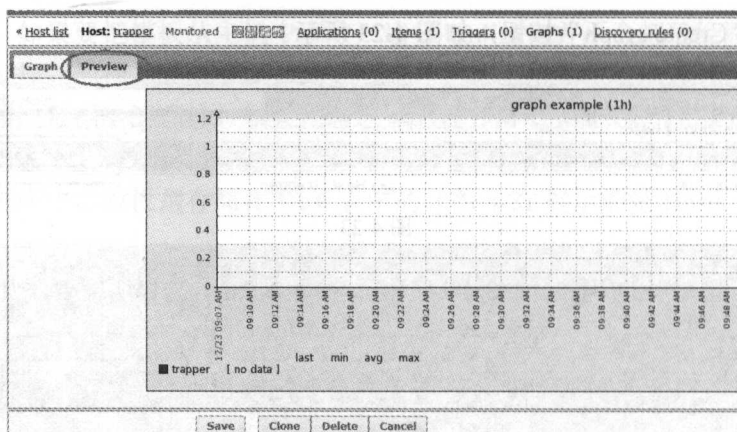


图 4-27

如果要查看所有的图形，单击 Monitoring→Graphs，如图 4-28 所示。

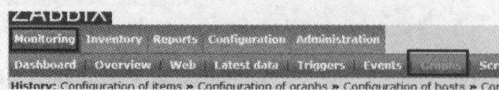


图 4-28

选择对应的 Group、Host、Graph，如图 4-29 所示。

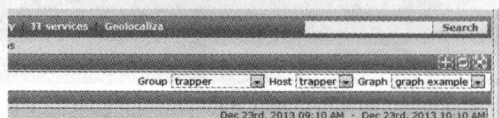


图 4-29

显示图形如图 4-30 所示。

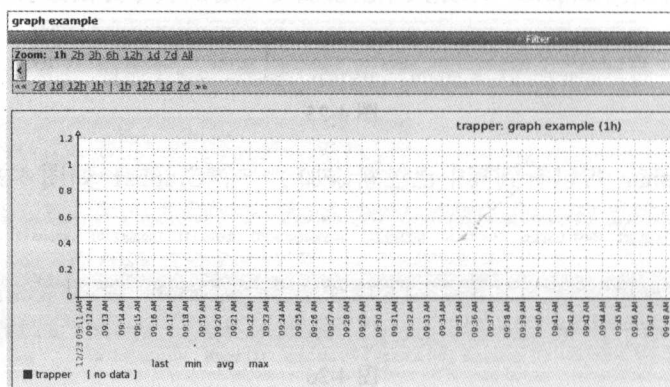
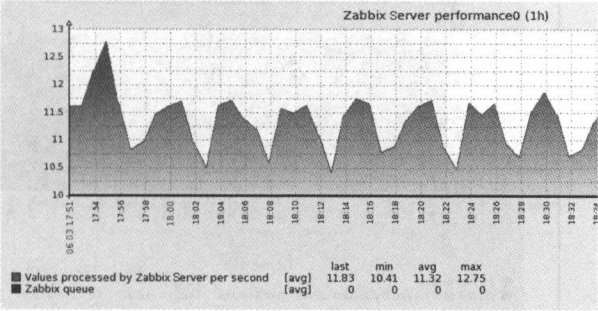
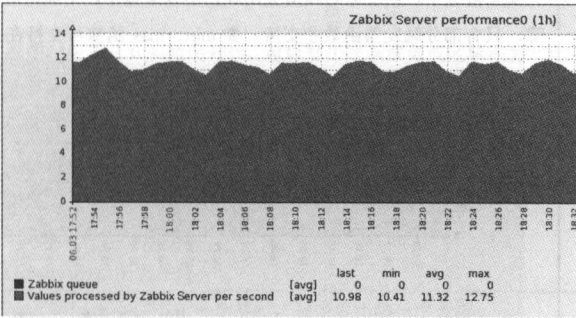
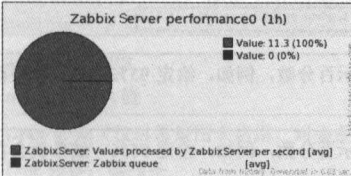
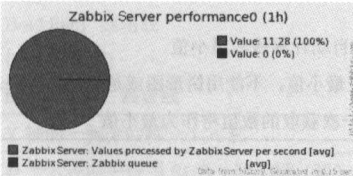


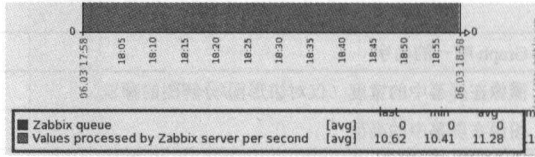
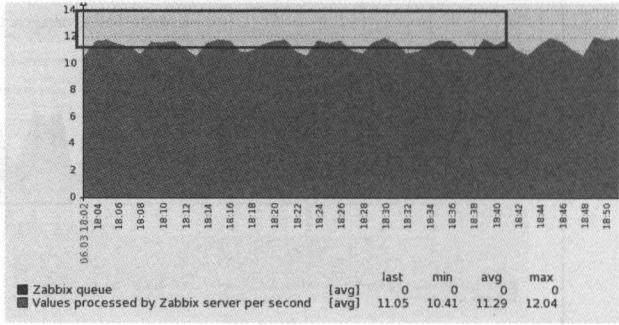
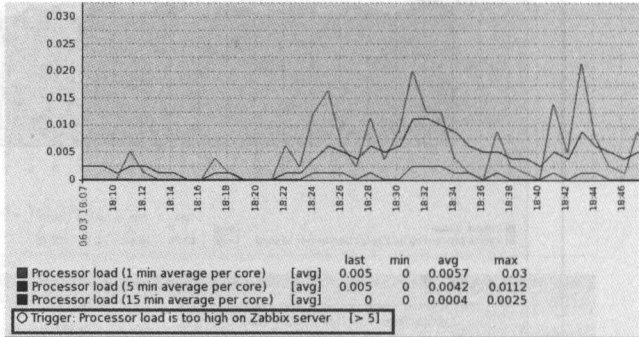
图 4-30

Graphs 的配置说明如表 4-2 所示。

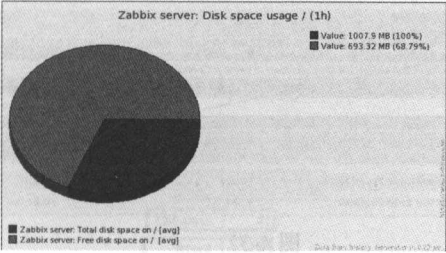
表 4-2

参 数	描 述
Name	Graph 唯一的名字
Width	图像在屏幕中的宽度（仅对饼形图/分解图的预览）
Height	图像在屏幕中的高度
Graph type	图表格式（图表类型）
	Normal: 常规图表，值用线条表示
	
	Stacked: 叠图
	
	Pie: 饼形图
	
	Exploded: 分解饼形图
	

续表

参 数	描 述
Show legend	<p>勾选会显示图表的说明</p>  <p>06.03.17.58 18.05 18.10 18.15 18.20 18.25 18.30 18.35 18.40 18.45 18.50 18.55 06.03.18.56</p> <p>■ Zabbix queue [avg] last min avg min ■ Values processed by Zabbix server per second [avg] 10.62 10.41 11.28 1</p>
Show working time	<p>如果被选中，非工作时间用灰色背景显示，不能用于饼形图或分解图中</p>  <p>06.03.18.02 18.04 18.06 18.08 18.10 18.12 18.14 18.16 18.18 18.20 18.22 18.24 18.26 18.28 18.30 18.32 18.34 18.36 18.38 18.40 18.42 18.44 18.46 18.48 18.50</p> <p>■ Zabbix queue [avg] last min avg max ■ Values processed by Zabbix server per second [avg] 11.05 10.41 11.29 12.04</p>
Show triggers	<p>如果被选中，触发达到阈值会用红色的线条显示，不能用饼形图或是分解图表示。注意，只有部分触发器函数才支持，如 min、max 函数可支持在图像中显示触发器的值</p>  <p>06.03.18.07 18.10 18.12 18.14 18.16 18.18 18.20 18.22 18.24 18.26 18.28 18.30 18.32 18.34 18.36 18.38 18.40 18.42 18.44 18.46</p> <p>■ Processor load (1 min average per core) [avg] last min avg max ■ Processor load (5 min average per core) [avg] 0.005 0 0.0057 0.03 ■ Processor load (15 min average per core) [avg] 0.005 0 0.0042 0.0112 ○ Trigger: Processor load is too high on Zabbix server [> 5]</p>
Percentile line (left)	<p>左边的 Y 轴用来显示百分数，例如，给定 95%，线条就会在 95% 的数值处，仅对常规图表适用</p>
Percentile line (right)	<p>右边的 Y 轴用来显示百分数，例如，给定 95%，线条就会在 95% 的数值处，仅对常规图表适用</p>
Y axis MIN value	<p>Y 轴表示的最小值 Calculated-Y 轴表示自动计算值的最小值 Fixed-Y 轴表示修正最小值，不能用饼形图或是分解饼形图表示 Item - items 的最后一次获取的数值将作为最小值</p>
Y axis MAX value	<p>Y 轴表示的最大值</p>

续表

参 数	描 述
3D view	3D 格式的图，仅对饼形图和分解饼形图适用 
Items	在这张图表中显示的 Item 获取值的数据

Items 的显示属性如图 4-31 所示，各参数说明如表 4-3 所示。

Name	Function	Draw style	Y axis side	Colour	Action
1: Zabbix server: Context switches per second	avg	Line	Right	009900	Remove
2: Zabbix server: Interrupts per second	avg	Line	Left	000099	Remove
Add					

图 4-31

表 4-3

参 数	描 述
Sort order (0→100)	画图时应该从 0 的顺序开始，可以被用来画线条或区域在另一个之后（或之前），可以在开始的线条箭头处拖放项目，以设定分类顺序或决定将哪一个项目放在另一个项目的前面
Name	Item 的名称显示的数据
Type	类型（仅对饼形图或是分解饼形图适用） Simple: 单一（简单） Graph sum: 图表总数
Function	当一个 Item 存在不止一个值时，决定显示哪一个数据 all: 全部（最小值、平均值和最大值） min: 仅最小值 avg: 仅平均值 max: 仅最大值
Draw style	Draw style（仅对常规图表适用，对叠图填充区域适用） Line 线条: 画线条 Filled region: 画填充区域 Bold line: 画粗线 Dot: 画圆点 Dashed line: 画虚线
Y axis side	Y 轴的一侧分给哪一个元素
Colour	在十六进制中应用 RGB（红、绿、蓝）法

在某些情况下,我们并不需要对所有的 Items 建立图像配置,这时可以通过 Simple Graphs 查看个别的数据图像,如图 4-32 所示。

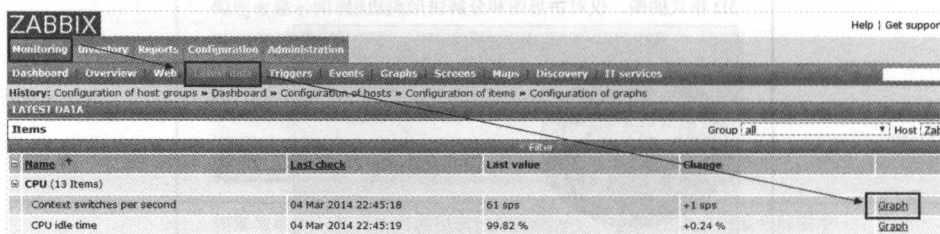


图 4-32

单击 Graph, 可以看到数据图, 如图 4-33 所示。

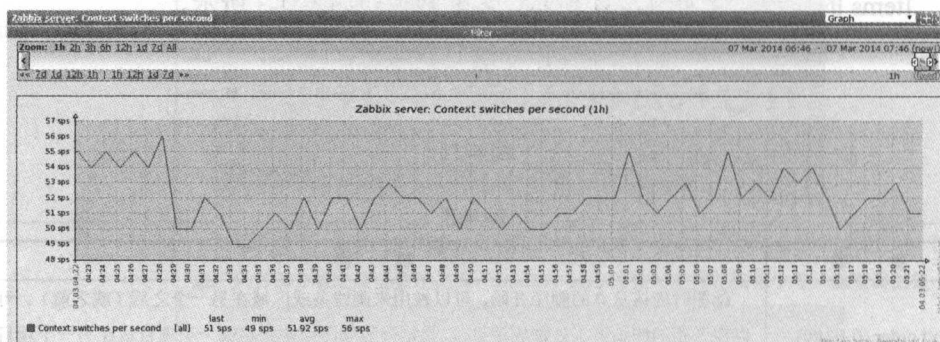


图 4-33

4.6 Screen 的配置

Screen 将多种信息放在一起展示,便于集中展示某个 Host 的多个信息,或是将多个 Hosts 的同一种信息放在一起显示,这些信息可以为 Graphs、Maps、Server info 等,几乎涵盖 Zabbix 所有的监控信息。

通过单击 Configuration→Screens→Creat screen 来创建,创建时定义 Screen 的行数和列数(如图 4-34 至图 4-36 所示),单击单元格内的 Change,添加相应的元素。

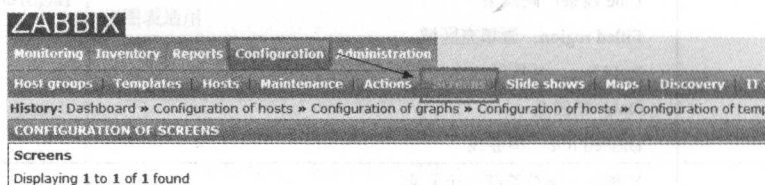


图 4-34

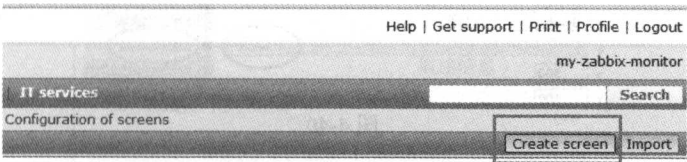


图 4-35

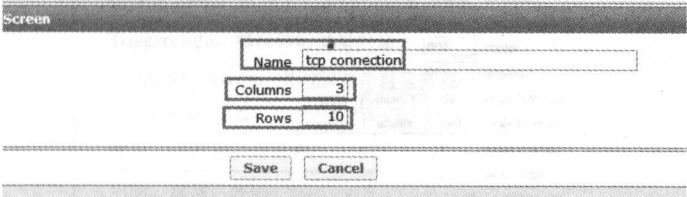


图 4-36

单击创建的 Screen 名称，如图 4-37 所示。

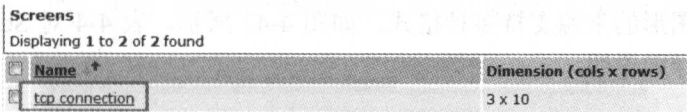


图 4-37

单击 Change，如图 4-38 所示。

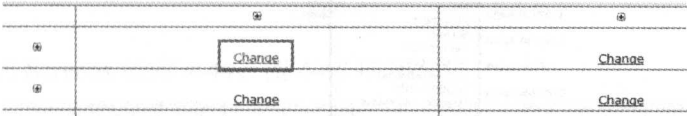


图 4-38

出现如图 4-39 所示的画面。

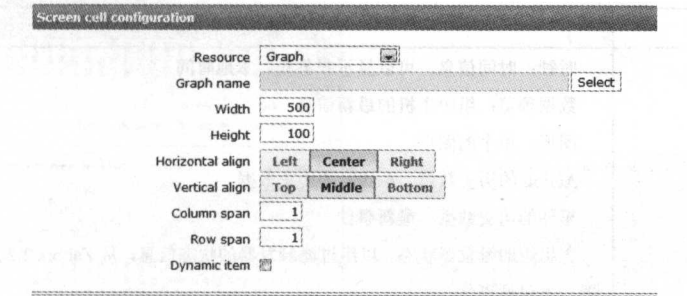


图 4-39

选择对应的 Graphs，这里选择的是 tcp connect 图形，如图 4-40 所示。

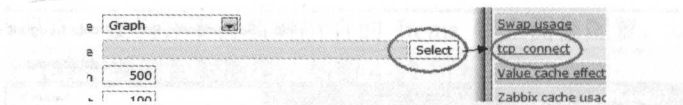


图 4-40

单击 Save 保存，如图 4-41 所示。

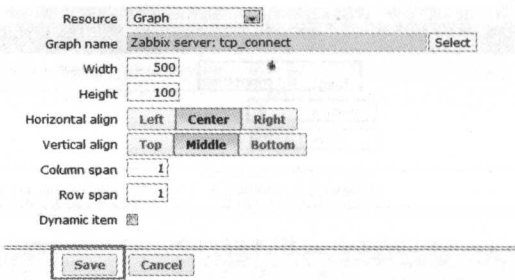


图 4-41

注意，图形的来源支持多种格式，如图 4-42 所示。表 4-4 对 Screens 的各种来源进行了详细说明。

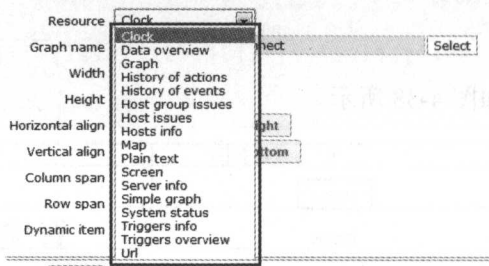


图 4-42

表 4-4

选 项	描 述 信 息
来源	<p>时钟：时间信息，可以显示服务器、本地时间</p> <p>数据预览：组中主机的最新信息</p> <p>图形：单个的图形</p> <p>Action 的历史数据：Action 的历史数据</p> <p>事件的历史数据：最新事件</p> <p>主机组的触发器状态：以组过滤触发器的状态信息，从 Zabbix 2.2 后，只包括触发器，不包括事件</p> <p>主机的信息：主机摘要信息</p> <p>地图：单个地图</p>

续表

选 项	描述信息
来源	文本信息：文本数据 Screen：一个或多个 Screen Server info：服务器的摘要信息 Simple graph：单个简单的图形，注意是没有添加到主机中的图形 System status：显示主机群组的状态 Triggers info：触发器的摘要信息 Triggers overview：组的触发器摘要信息 URL：URL 地址
水平对齐方式	居中、左对齐、右对齐
垂直对齐方式	中间、顶部、底部
列跨幅	一个图形占用几列
行跨幅	一个图形占用几行

配置好后，通过单击 Monitor→Screens，选择相应的 Screens 组，如图 4-43 所示。

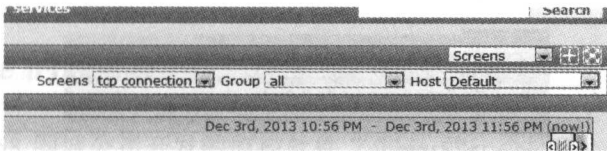


图 4-43

添加好的 Screens 如图 4-44、图 4-45 所示。

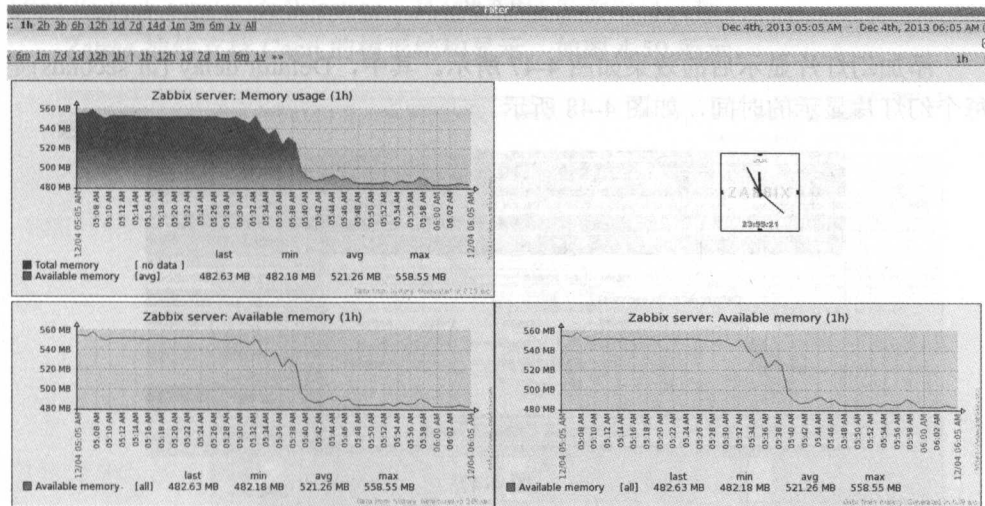


图 4-44

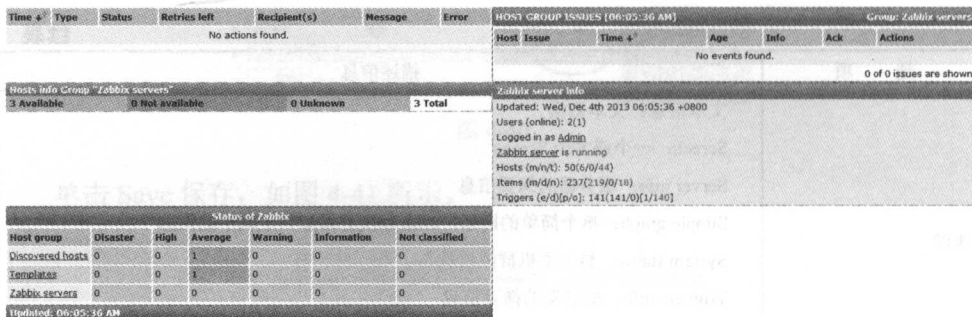


图 4-45

4.7 Slide shows 的配置

Slide shows 在多个 Screens 之间以幻灯片的方式来展示，这样一个屏幕就可以显示多个页面了。

在配置 Slide shows 之前，需要先配置 Screens，图 4-46 是配置了 3 个 Screens 的例子。

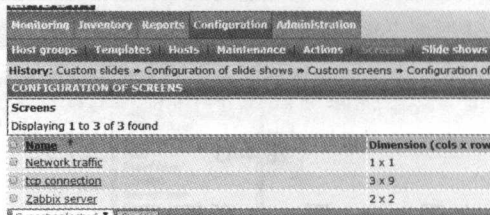


图 4-46

添加幻灯片显示后的效果如图 4-47 所示。其中，Default delay (in seconds)是每个幻灯片显示的时间，如图 4-48 所示。

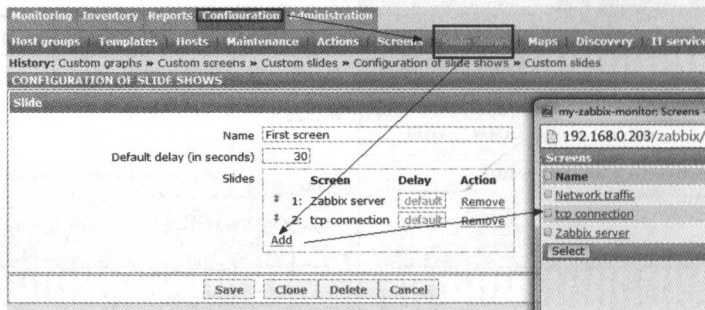


图 4-47

单击 Monitoring→Screens→Slide shows，查看幻灯片显示情况，如图 4-49 所示。

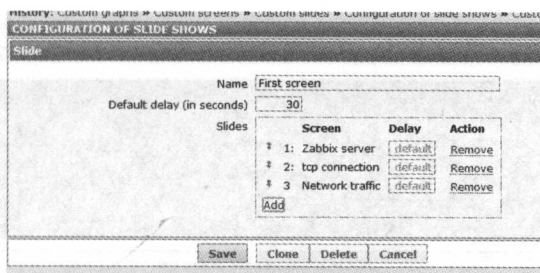


图 4-48

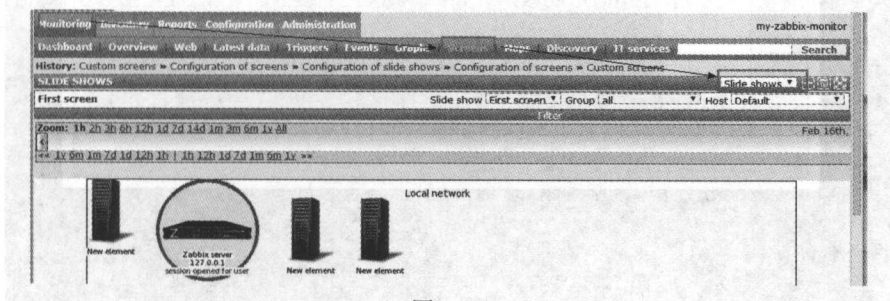


图 4-49

4.8 Zatree 的使用

Zatree 是 58 同城公司开发的监控软件 Zabbix 的一个插件，主要功能是提供 Host group 的树形展示和在 Item 中指定关键字查询及数据排序。

该项目由@南非蜘蛛、@千里笨笨、@Jason 阿坚（微博）发起。项目地址为 <https://github.com/spide4k/zatree>，具体配置请参考项目文档。

Zatree 插件支持 Cacti 的树形结构显示，如图 4-50 所示。

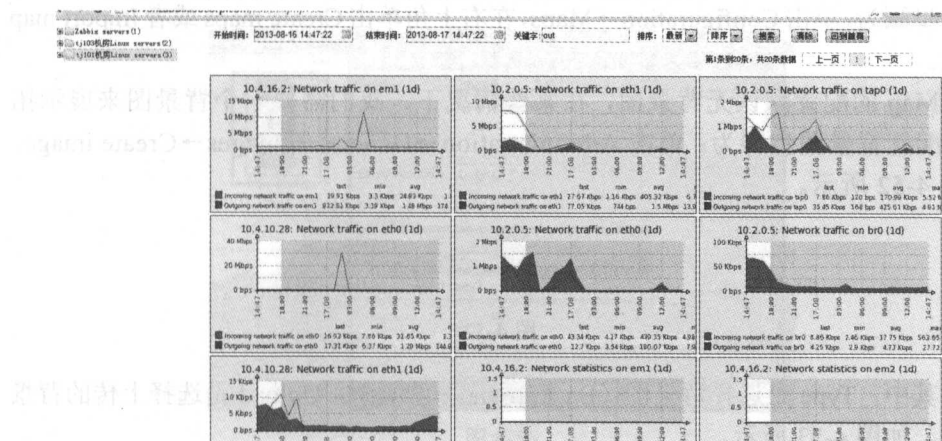


图 4-50

也支持单个图形的放大显示，如图 4-51 所示。

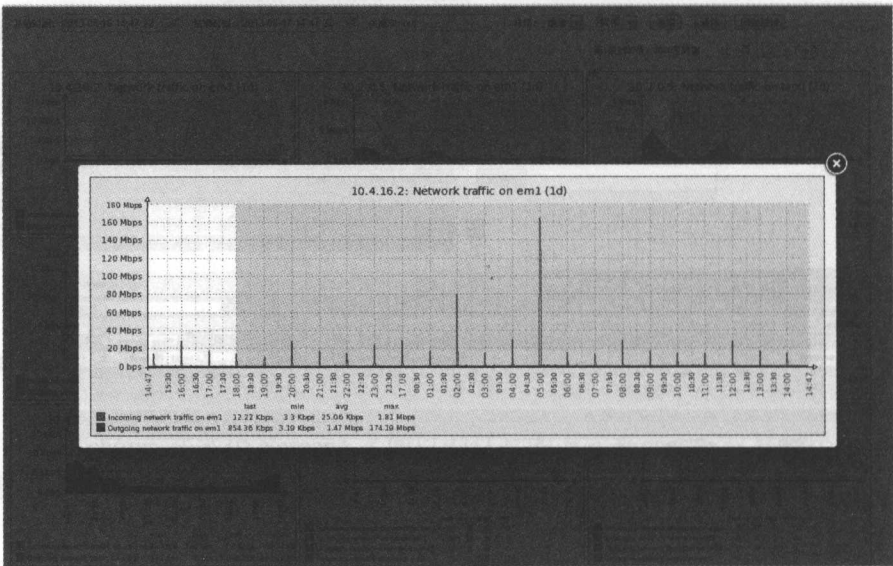


图 4-51

Zatree 插件的这个项目弥补了 Zabbix 在图形显示上的功能缺陷，感谢该项目为 Zabbix 的易用所做出的努力。

4.9 Map 的配置

Map 的作用是将各种设备用网络拓扑图的方式展示，在 Zabbix 中，这种拓扑图的展示通过手动方式添加。

步骤为：单击 Configuration→Maps，在右上角单击 Create maps 或者 Import map 完成。

Map 的配置默认无背景图，在某些情况下，我们需要一个背景图来展示拓扑，配置背景图步骤为：单击 Administration→General→Images→Create image，如图 4-52 所示。

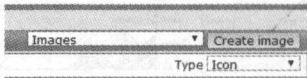


图 4-52

其中，Type 是选择背景图（Background）的类型；Upload 是选择上传的背景图片，如图 4-53 所示。

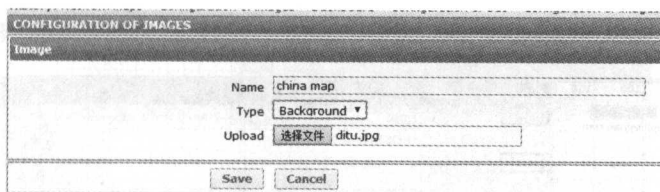


图 4-53

使用背景图的方法如下：

单击 Configuration→Maps→Create map→Background image，选择自定义的背景图片，如图 4-54 所示。

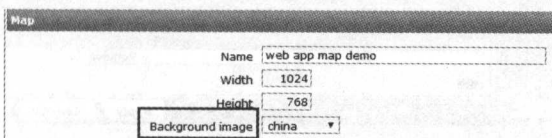


图 4-54

添加 Map 中的主机，单击“+”号按钮添加一个设备，如图 4-55 所示。

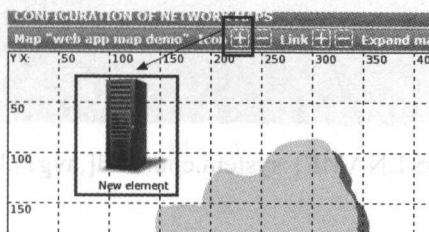


图 4-55

单击新添加的设备，弹出编辑选项，如图 4-56 所示。

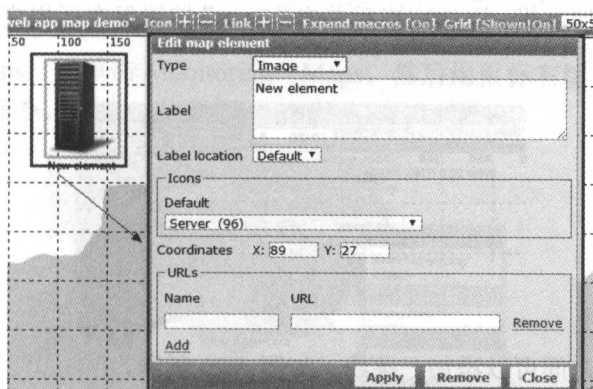


图 4-56

改变图 4-56 中的元素设置，这里以添加一个 Host 为例，如图 4-57 所示。

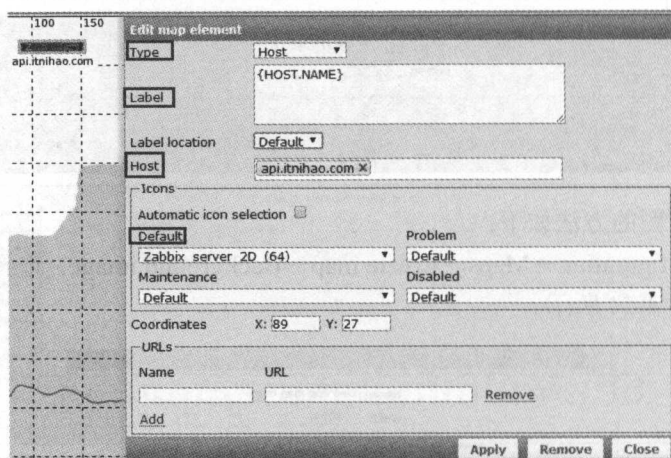


图 4-57

其中，Label 文本框中可以使用宏，可用的宏如下。

```
{HOST.NAME}
{HOST.CONN}
{HOST.DNS}
{HOST.IP}
{HOST.HOST}
```

例如，添加宏 `{{HOST.NAME}:system.cpu.load[,avg1].last(0)}`，可以显示 CPU 负载。

其他的宏参数请参考如下地址。

https://www.zabbix.com/documentation/2.2/manual/appendix/macros/supported_by_location

依次添加其他主机，添加完毕后单击“Save”按钮保存后提交更改，如图 4-58 所示。

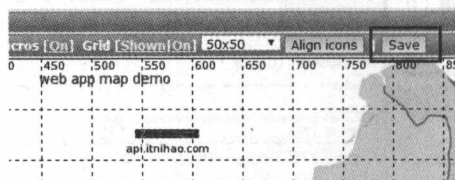


图 4-58

在两个设备之间可以添加一条线，如图 4-59 所示。

单击“Edit”按钮进行编辑，在 Label 中可以添加两条连线之间的标签，这个标签中还可以添加宏，如图 4-60 所示。

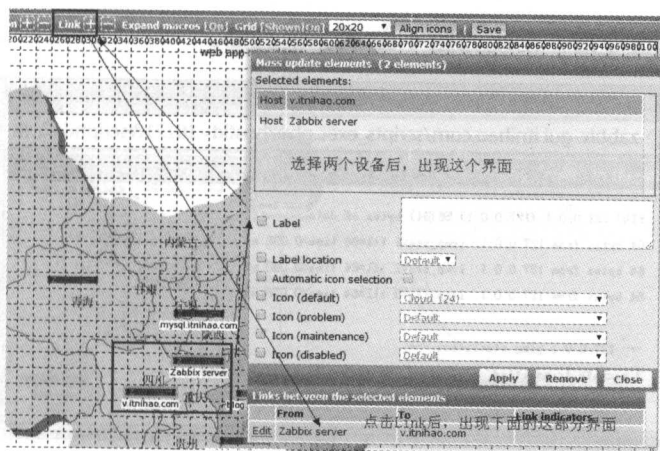


图 4-59

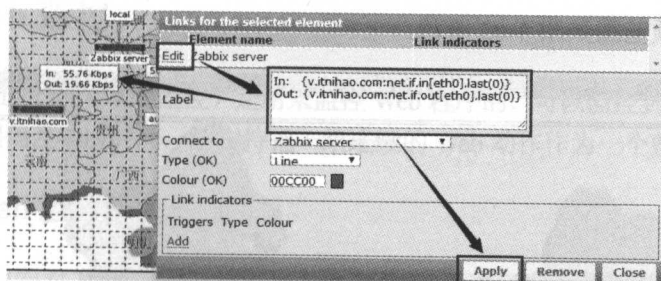


图 4-60

宏的设置如下。

```
In: {v.itnihao.com:net.if.in[eth0].last(0)}
Out: {v.itnihao.com:net.if.out[eth0].last(0)}
```

其中，v.itnihao.com 是主机，net.if.in[eth0]是 key，last(0)是最后一次的值，这样就可以显示实时的值了。

若要查看 maps，可单击 Monitoring→Maps，然后用鼠标右键单击某个设备，执行相关的操作命令，如 Ping，如图 4-61 所示。

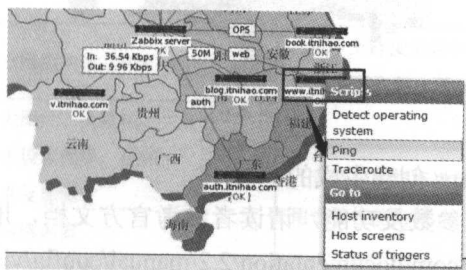


图 4-61

执行 Ping 命令后, 出现如图 4-62 所示的界面。

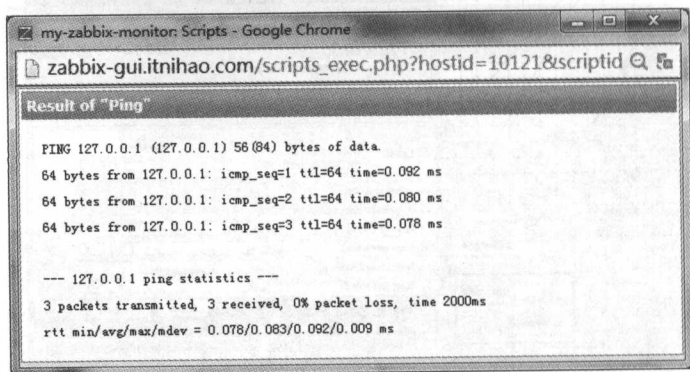


图 4-62

Maps 的预览图如图 4-63 所示。

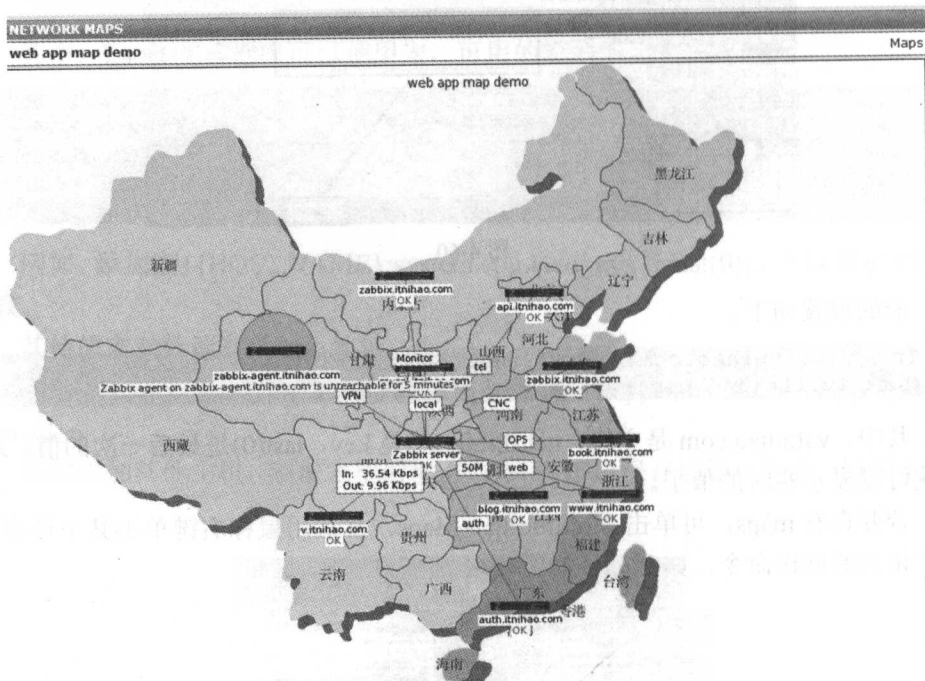


图 4-63

图 4-64 也是由 Maps 创建生成的。

有关 Maps 更多的参数及功能，请读者参考官方文档，地址如下。

<https://www.zabbix.com/documentation/2.2/manual/config/visualisation/maps/map>

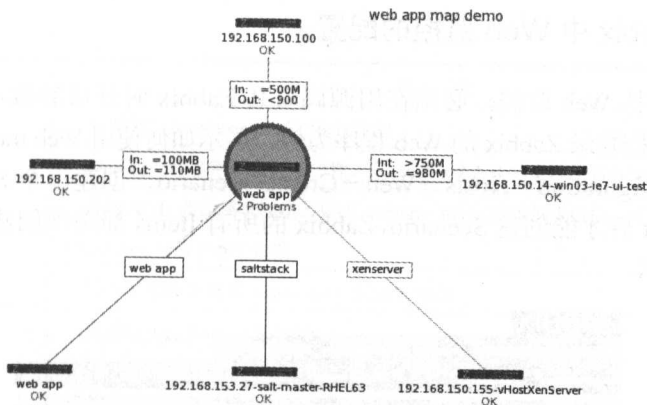


图 4-64

4.10 Web 监控

Web monitoring (Web 监控) 是用来监控 Web 程序的，可以监控到 Web 程序的下载速度、返回码及响应时间，还支持把一组连续的 Web 动作作为一个整体进行监控。

4.10.1 Web 监控的原理

Web 监控即对 HTTP 服务的监控，模拟用户去访问网站，对特定的结果进行比较，如状态码、返回字符串等特定的数据进行比较和监控，从而判断网站 Web 服务的可用性。在很多时候，我们可以用脚本、程序来进行自定义监控，如 Linux 下的命令 curl、http 库等多种现有的程序和库都可以帮我们完成这一需求。

4.10.2 Web 监控指标

Web 监控指标如表 4-5 所示。

表 4-5

监控项目	特征	说明
HTTP 状态码	重点监控 40X、50X	404 空链接是影响性能的一个重要指标，50X 表示服务器内部出现问题
HTTP 响应速度	<ul style="list-style-type: none">大图片、大视频、大文件未设置缓存，压缩重复加载网络因素服务器性能	对特定的指标进行抽样监控，及时发现服务的可用性和性能指标
HTTP 下载速度		对特定的文件抽样下载

4.10.3 Zabbix 中 Web 监控的配置

如果要支持 Web 监控，必须在用源码安装 Zabbix 时开启参数--with-libcurl。

下面以监控登录 Zabbix 的 Web 程序为例，展示如何使用 Web monitoring。

单击 Configuration→Hosts→Web→Create scenario，创建一个 Scenario（注：必须选择 Host 后才能创建 Scenario，Zabbix 的所有 Items 都必须创建在 Hosts 上），如图 4-65 所示。

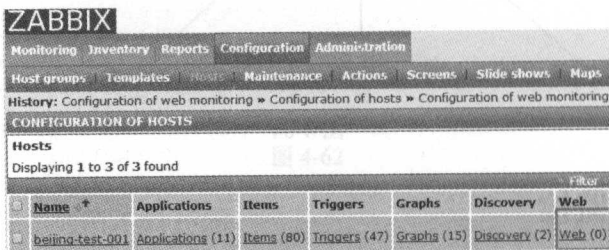


图 4-65

单击图 4-64 中的 Web，出现如图 4-66 所示的界面。

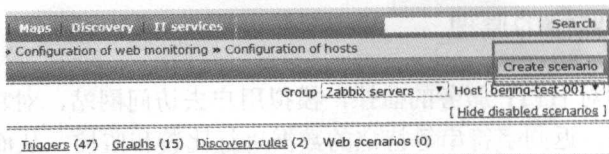


图 4-66

单击 Create scenario，出现如图 4-67 的界面，其中的各参数及说明如表 4-6 所示。

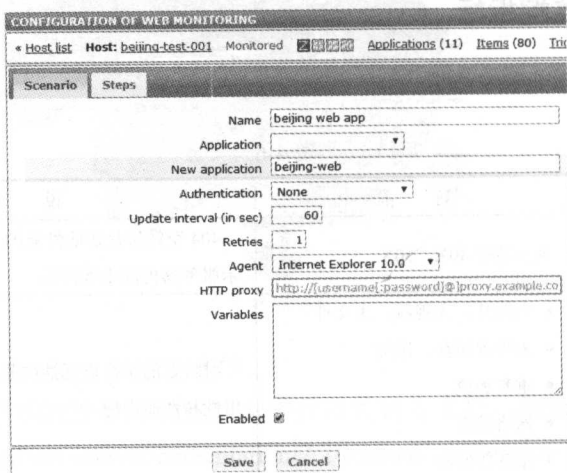


图 4-67

表 4-6

参 数	说 明
Host	配置属于 host/template 的 Web 监控
Name	Web 监控的名字，具有唯一性。从 Zabbix 2.2 开始支持在名称中使用宏
Application	选择 Web 监控属于的组。Web scenario items 配置属于组后，可以在 Monitoring→Latest data 分组中看到数据
New application	输入名称即为场景（scenario）创建新的组
Authentication	支持的认证选项，可选值如下。 None：无认证 Basic authentication：基本认证 NTLM authentication：NTLM（Windows NT LAN Manager）认证 选择认证后，需要填入用户名和密码。从 Zabbix 2.2 开始，用户名和密码选项支持使用宏
Update interval (in sec)	Scenario 间隔的时间，单位是秒
Retries	重试机制，默认是 1，最多支持 10 次重试，从 Zabbix 2.2 开始支持此参数
Agent	浏览器的类型，支持自定义。从 Zabbix 2.2 开始支持使用宏
HTTP proxy	HTTP 代理格式 http://[username[:password]@]proxy.mycompany.com[:port] 默认端口使用 1080。 注意：只是简单的认证才支持 HTTP proxy；Zabbix 2.2 以上版本支持此参数
Variables	Scenario 级的变量（macros）可以在 scenario steps (URL, Post variables)中配置，格式如下： {macro1}=value1 {macro2}=value2 {macro3}=regex:<regular expression> 例如： {username}=Alexei {password}=kj3h5kJ34bd {hostid}=regex:hostid is ([0-9]+) 如果值中含有正则表达式，正则表达式匹配会从 Web 页面中搜索，如果找到，则将变量替换为对应的值 从 Zabbix 2.2 开始，支持 HOST.* 宏和用户自定义的宏
Enabled	Scenario 开启关闭

Steps 表示可以按步骤设置多个监控项，如图 4-68 是添加 Steps 的监控项。

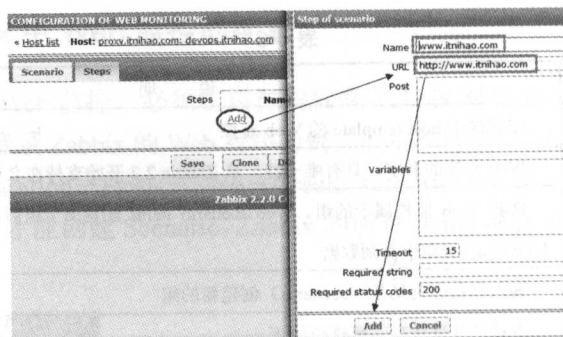


图 4-68

其中，各参数的含义如下。

- URL: 监控的 Web 页面（注：必须是全路径带页面名）。
- Post: 传递给页面的参数，多个参数之间用&连接，此处可引用前面定义的变量。
- Variables: 设置变量。
- Timeout: 超时时间。
- Required: 页面中能匹配到字符，若不匹配，则认为出错。
- Status codes: 页面返回码。

如果有多个 URL，依次添加，如图 4-69 所示，图形界面如图 4-70、图 4-71 所示。

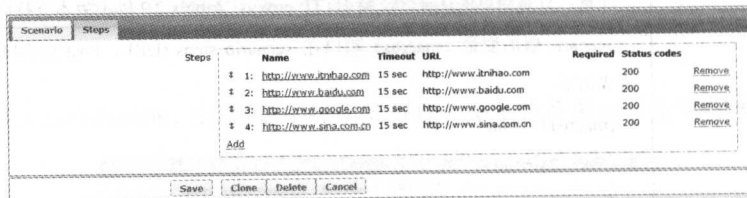


图 4-69

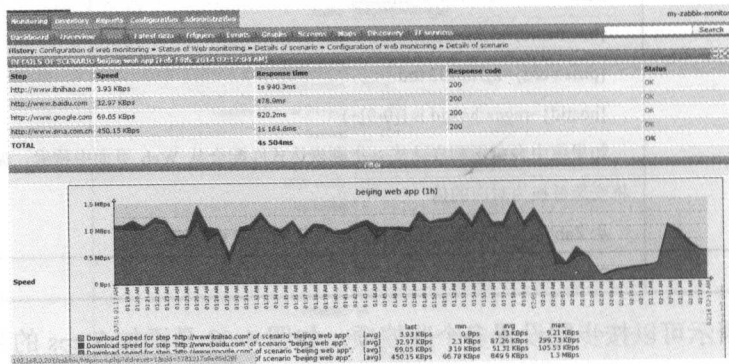


图 4-70

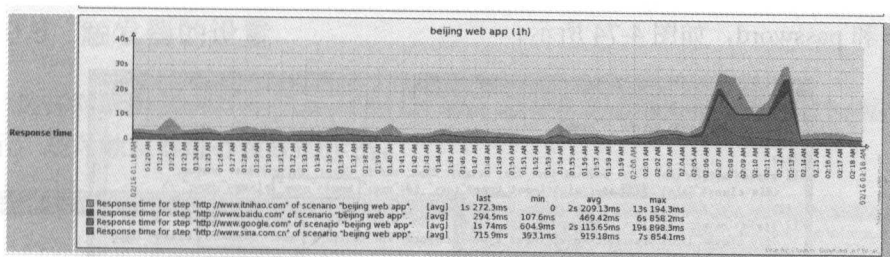


图 4-71

4.10.4 认证的支持

支持用户名和密码的 Web 页面监控如图 4-72 所示。

图 4-72

如图 4-73 所示的一个 Web 页面采取了认证的方式，这里可以在 Web 监控中对用户名和密码设置变量。

图 4-73

在这两个 HTTP Post 数据的网页源码中，可以看到用户名和密码的变量为

name 和 password，如图 4-74 所示。

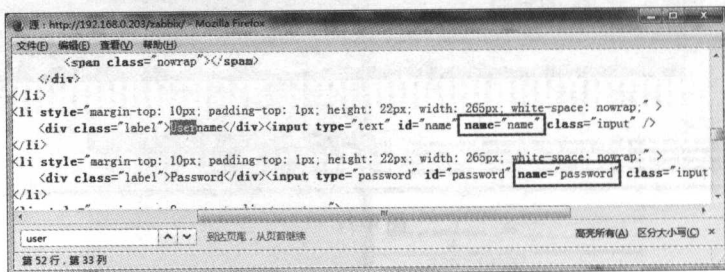


图 4-74

因此，Post 框中的设置和变量的设置如图 4-75 所示。

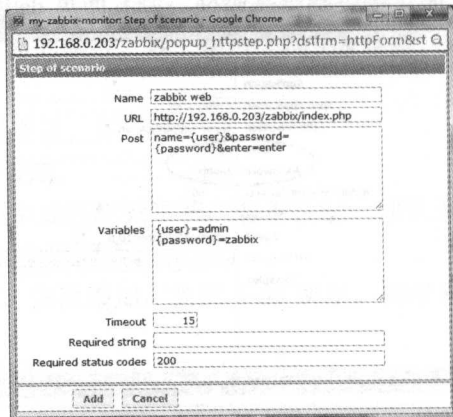


图 4-75

添加好后的监控数据如图 4-76 所示。

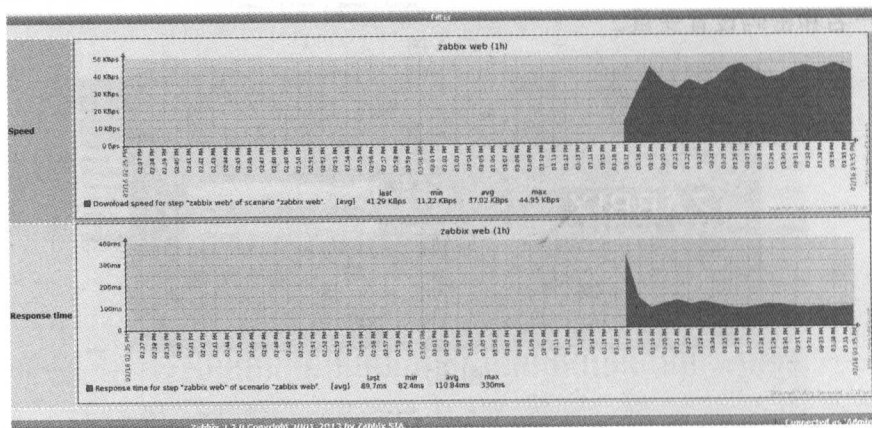


图 4-76

4.10.5 触发器的设置

选择对应的 Host/Template，然后对 Web 监控创建触发器，如图 4-77、图 4-78 所示，是对触发器进行设置的示例。

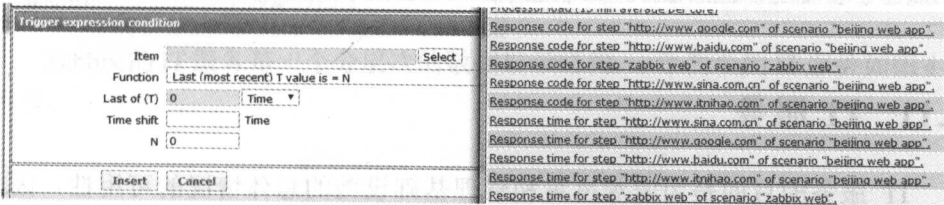


图 4-77

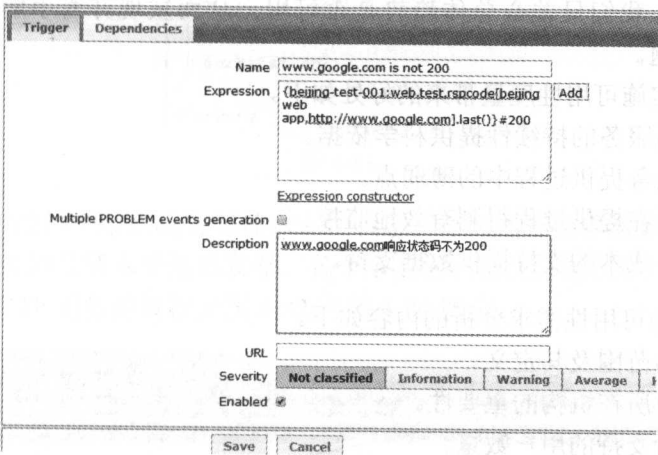


图 4-78

关于如何设置触发器，请参考具体的触发器配置的内容。
到此，一个完整的 Web monitoring 就配置完成了。

4.10.6 排错

如果配置出现错误，Web 监控会给出相关的提示，如图 4-79、图 4-80 所示，均是错误提示信息。

Monitoring Inventory Reports Configuration Administration				
Dashboard	Overview	Web	Latest data	Triggers
History: Status of Web monitoring » Dashboard » Configuration of web monitoring » Status of Web monitoring » Details of scenario				
DETAILS OF SCENARIO beijing web app [Feb 16th, 2014 12:44:28 AM]				
Step	Speed	Response time	Response code	Status
http://www.itnihao.com	3.45 KBps	2s 210.8ms	200	Error: status code did not match
TOTAL		2s 210.8ms		Error: status code did not match

图 4-79

DETAILS OF SCENARIO beijing web app [Feb 16th, 2014 04:12:34 PM]				
Step	Speed	Response time	Response code	Status
http://www.itnihao.com -	-	-	-	Error: Couldn't resolve host name
http://www.baidu.com -	-	-	-	Unknown
http://www.google.com -	-	-	-	Unknown
http://www.sina.com.cn -	-	-	-	Unknown
TOTAL	-	-	-	Error: Couldn't resolve host name

图 4-80

4.11 IT 服务

IT 服务的目的是为宏观度量和管理基础设施的总体情况的可用性，因为在很多时候，我们无须关心基础设施的细节问题，这包括磁盘空间不足、处理器高负载等，我们只关心总体趋势是否可用，从而发现并着手解决 IT 基础设施暴露的问题。

IT 服务实施可用性度量带来的好处如下。

- 对提高服务的持续性提供科学依据。
- 明确服务提供过程中的薄弱点。
- 使服务在提供过程得到有效地监控。
- 为服务成本的支持提供数据支持。

IT 服务的可用性需求分析的内容如下。

- 服务的范围及其定义。
- 服务对所在机构的重要性。
- 服务所支持的用户数量。
- 服务不可用状态造成的业务影响程度。
- 服务不可用状态产生的成本代价以及成本代价随时间推移，相应变化的趋势。
- 服务提供的可用时间。
- 服务提供的关键时段，如业务的高峰、月末及处理期限等。
- 服务提供次要时段，如较容易承受停机故障的时间。
- 服务计划中断时段，如专为计划内维护与系统升级工作设定的停机时间。
- 服务容许的不可用时间，如启动应急计划之前所能容许的故障停机时间。

IT 服务的要求级别如表 4-7 所示。

表 4-7

可用性分类	可用性级别	每月服务中断时间	每年服务中断时间
持续性	100%	0 分钟	0 分钟
容错性	100.00%	0.42 分钟	5 分钟

续表

可用性分类	可用性级别	每月服务中断时间	每年服务中断时间
弹性（冗余）	99.99%	4.42 分钟	53 分钟
高可用性	99.90%	44 分钟	528 分钟（8.8 小时）
一般可用性	99%~99.5%	438~219 分钟	87.6~43.8 小时

Zabbix 的 IT 服务是一个分层次的数据展示结构，一个简单的 IT 服务如图 4-81 所示。

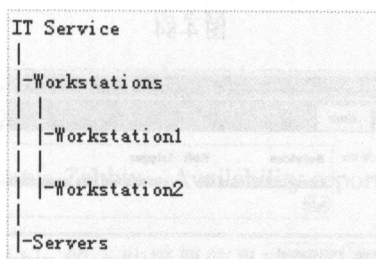


图 4-81

该结构中的每个节点具有属性状态，状态根据所选择的算法计算并传播到上层。在 IT 服务的最低水平是触发器。各个节点的状态受其触发器所影响。

创建一个 IT 服务的过程如图 4-82 至图 4-86 所示。

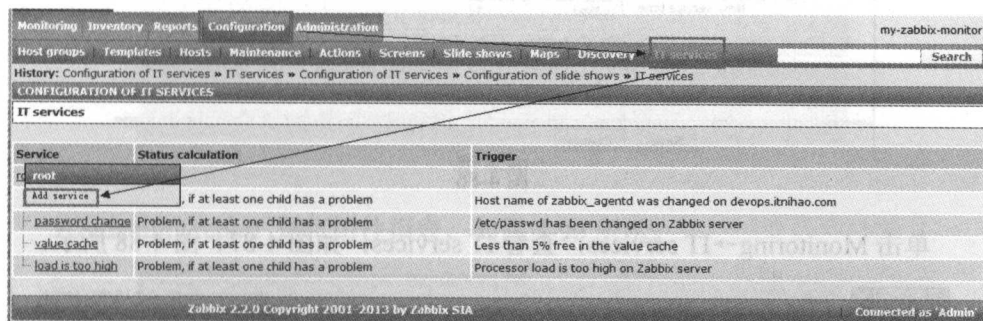


图 4-82

Service	Status calculation
root	
hostname	least one child has a problem
Add service	least one child has a problem
Edit service	least one child has a problem
Delete service	least one child has a problem

图 4-83

Monitoring Inventory Reports Configuration Administration

Host groups Templates Hosts Maintenance Actions Screens Slide shows Maps Discovery IT services

History: Configuration of IT services » Configuration of slide shows » IT services » Configuration of IT services » IT services

CONFIGURATION OF IT SERVICES

Service Dependencies Time

Name

Parent service root

Status calculation algorithm Problem, if at least one child has a problem

Calculate SLA, acceptable SLA (in %) 99.00

Trigger

Sort order (0->999) 0

图 4-84

CONFIGURATION OF IT SERVICES

Service Dependencies Time

Depends on

Services	Soft	Trigger	Action
password change	<input checked="" type="checkbox"/>	/etc/passwd has been changed on Zabbix server	<input type="button" value="Remove"/>
<input type="button" value="Add"/>			

图 4-85

Service Dependencies Time

Service times

Type	Interval	Note	Action
No times defined. Work 24x7.			

New service time

Uptime

From Sunday Time hh : mm

Till Sunday Time hh : mm

图 4-86

单击 Monitoring→IT services，查看 IT services，如图 4-87、图 4-88 所示。

ZABBIX

Help | Get support | Print | Profile | Logout

Monitoring Inventory Reports Configuration Administration my-zabbix-monitor

Dashboard Overview Web Latest data Triggers Events Graphs Screens Maps Discovery IT services

History: Configuration of IT services » IT services » Configuration of IT services » Configuration of slide shows » IT services

IT SERVICES

IT services Period Last 7 days

Service	Status	Reason	Problem time	SLA / Acceptable SLA
root				
hostname-Host name of zabbix_agentd was changed on devops.tnshao.com	OK	-	0.0000	100.0000 / 99.9000
password change-/etc/passwd has been changed on Zabbix server	OK	-	0.0000	100.0000 / 99.9000
value cache-Less than 5% free in the value cache	OK	-	0.0000	100.0000 / 99.9000
load is too high-Processor load is too high on Zabbix server	OK	-	80% 100% 0.0000	100.0000 / 99.9000

Zabbix 2.2.0 Copyright 2001-2013 by Zabbix SIA

Only the last 20% of the indicator is displayed.

connected as Admin

图 4-87

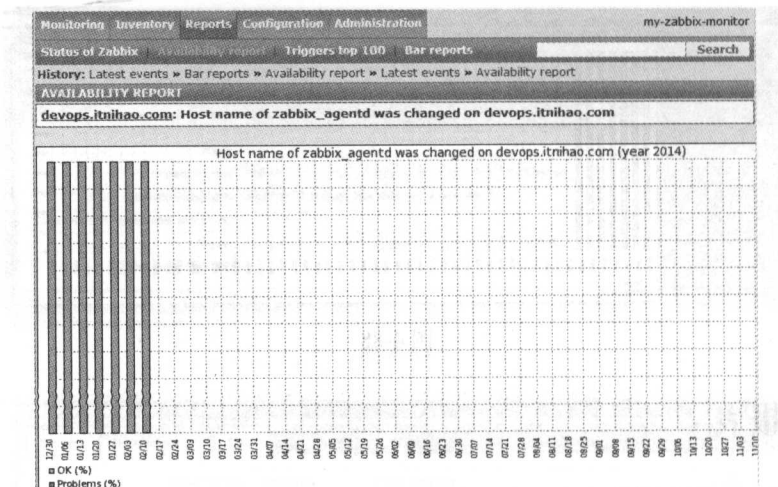


图 4-91

通过 Triggers top 100 显示前 100 个触发器信息，如图 4-92 所示。

Monitoring

Inventory

Reports

Configuration

Administration

my-zabbix-monitor

Status of Zabbix

Availability report

Triggers top 100

Bar reports

Search

History: Latest events » Bar reports » Availability report » Latest events » Availability report

MOST BUSY TRIGGERS TOP 100

Report

Day

Host	Trigger	Severity	Number of status changes
Zabbix server	User login number gt 2 on Zabbix server	Warning	2
Zabbix server	Zabbix server has just been restarted	Information	2
Zabbix server1	User login number gt 2 on Zabbix server1	Warning	2
Zabbix server1	Zabbix server1 has just been restarted	Information	2
Zabbix server2	User login number gt 2 on Zabbix server2	Warning	2
Zabbix server2	Zabbix server2 has just been restarted	Information	2

图 4-92

定制报表内容的方法如图 4-93 所示。

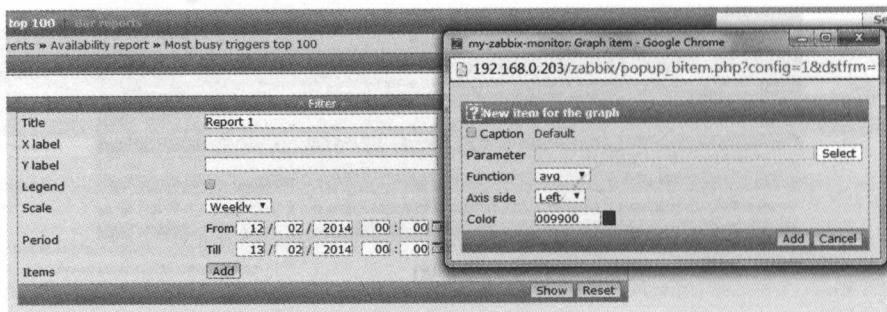


图 4-93

定制报表的内容如图 4-94 所示。

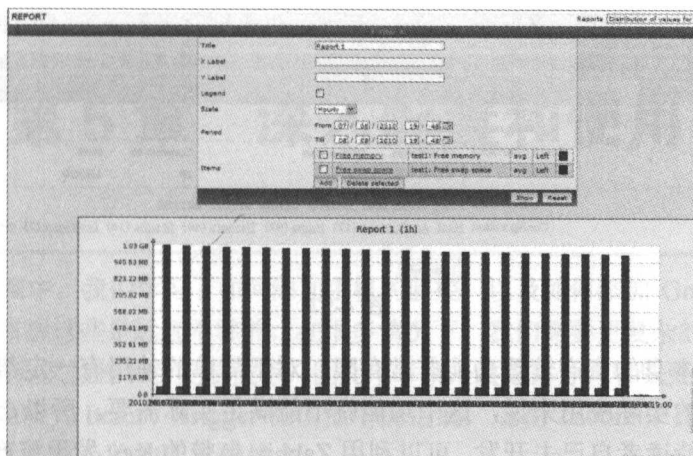


图 4-94

Zabbix 的报表仅能呈现一些基本的图形数据，不能自动生成文本的报告，如果有这方面的需求，可以用 API 手动编写程序生成报表数据，然后自动发送到邮箱。

4.13 资产管理

配置资产管理是在 Host 下面完成的，该项默认是关闭的。也可以自动获取，或者手工输入，如图 4-95 所示。

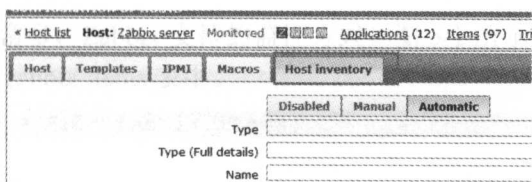


图 4-95

在资产管理菜单栏可以看到相关的资产设备，如图 4-96 所示。

Monitoring	Inventory	Reports	Configuration	Administration
Overview	Host inventory overview	Host inventory	Configuration of hosts	Host inventory
History: Host inventory overview » Host inventory » Configuration of hosts » Host inventory » Host inventory				
HOST INVENTORY				
Hosts				
Displaying 1 to 1 of 1 found				
Host	Group	Name	Type	OS
Zabbix server	Zabbix servers			

图 4-96

查看到的资产如图 4-97 所示。

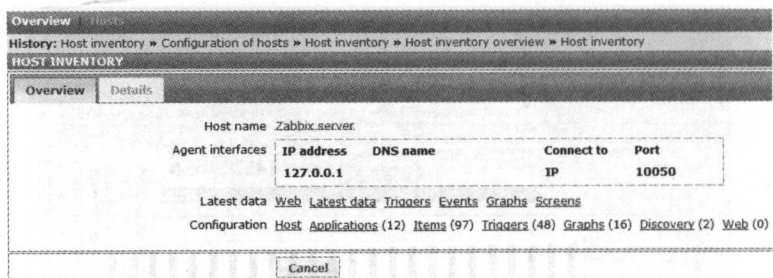


图 4-97

Zabbix 本身的资产管理功能非常有限，对于简单的使用有一定帮助，但对于有更多需求的实际应用来说，这个功能就不能满足实际需要。所以，真正的资产管理工具有待读者自己去开发，可以利用 Zabbix 自带的 Key 发现资产参数，将其保存到自己的资产管理系统 CMDB 中。

第 5 章 深入配置和使用

在第 4 章中，我们学习了如何添加主机的监控，以及如何添加 Graphs、Screen 和 Map。仅掌握这些是远远不够的，在很多情况下，我们需要更复杂的监控方式，此时就需要自定义监控和自定制模板，因此本章就是对这部分内容做更深入的研究。

本章主要研究 Items、Trigger 在模板中的定制，这在 Zabbix 中是非常重要的内容，可以说是整个监控配置的核心。

5.1 Items 的添加

5.1.1 Items 的含义

Items 就是监控项可以配置获取监控数据的方式、取值的数据类型、获取数值的间隔、历史数据保存时间、趋势数据保存时间、监控 Key 的分组等。

检测周期的长短直接影响了数据的获取，也影响了 Zabbix Server 的性能。

监控的方式非常多，Zabbix 支持的监控方式如图 5-1 所示。

监控项存在于 zabbix.items 表中。



图 5-1

```
mysql> select * from zabbix.items;
```

5.1.2 如何添加 Items

Items 可以存在于模板（Template）中，也可以存在于主机（Host）中，模板的作用是可以复用，对需要重复配置的监控项归类。

下面演示如何在 Template OS Linux 模板中添加一个 Items，Key 为 vm.memory.size[total]。

Template OS Linux 模板中默认不存在统计总内存大小的 Items，故我们需要自己添加，添加方法为：单击 Configuration→Templates，如图 5-2 所示。

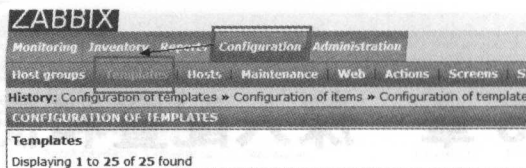


图 5-2

找到 Template OS Linux，单击 Items，如图 5-3 所示。

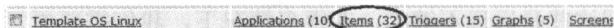


图 5-3

单击 Create item 创建 Item，如图 5-4 所示。

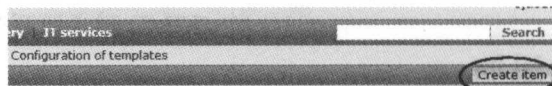


图 5-4

单击 Select 按钮，列出 Agent 所有自带的 Key，如图 5-5 所示。由于 vm.memory.size[] 这个 Key 是 Zabbix 自带的，所以会列出来。这里需要注意，如果是自己定义的 Key，单击 Select 是无法看到的。自己定义的 Key 在这里手动填入即可。

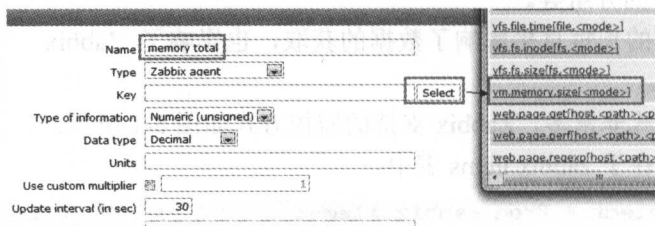


图 5-5

单击 `vm.memory.size[<mode>]` 后，会自动填充到 Key 的文本框中，变成如图 5-6 所示的形式。

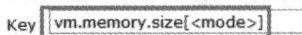


图 5-6

图 5-6 中自动填充的是一个默认的 Key，方括号[]中是可选的参数，这里需要自己去输入，我们的目的是获取总的内存大小，通过查看 Agent 文档（地址为 https://www.zabbix.com/documentation/2.2/manual/config/items/itemtypes/zabbix_agent）知道，获取总的内存大小的参数是 Total。

下面看一下官方文档中对 `vm.memory.size[]` 的解释，如图 5-7 所示。

vm.memory.size[<mode>]			Old naming: <code>vm.memory.buffers</code> , <code>vm.memory.cached</code> , <code>vm.memory.free</code> , <code>vm.memory.shared</code> , <code>vm.memory.total</code>
Memory size	Memory size in bytes or in percentage from total	mode - one of total (default), active, anon, buffers, cached, exec, file, free, inactive, pinned, shared, wired, used, pused, available, pavailable	Item <code>vm.memory.size[]</code> accepts three categories of parameters. First category consists of total - total amount of memory. Second category contains platform-specific memory types: active, anon, buffers, cached, exec, file, free, inactive, pinned, shared, wired . Third category are user-level estimates on how much memory is used and available: used, pused, available, pavailable . See a more detailed description of <code>vm.memory.size</code> parameters.

图 5-7

我们需要 Total 参数，所以改变 Key 的值为 `vm.memory.size[total]`，如图 5-8 所示。

Key `vm.memory.size[total]`

图 5-8

如何确认我们的 Key 添加正确呢？当然是有测试的方法的，用 `zabbix_get` 来获取值，语句如下。

```
shell# zabbix_get -s 127.0.0.1 -k vm.memory.size[total]
961351680
```

确保能获取到值，就说明 Key 配置是正确的。这里再次提醒，`zabbix_get` 仅能测试获取 Agent 监控方式的 Key 值，不能获取 Simple Check、JMX、SNMP 等其他监控方式的 Key 类型的数据。

最终添加的结果如图 5-9 所示。

Item

Name

memory total

Type

Zabbix agent

Key

vm.memory.size[total]

Select

Host interface

127.0.0.1 : 10050

Type of information

Numeric (unsigned)

Data type

Decimal

Units

Use custom multiplier

☐

Update interval (in sec)

30

Flexible intervals

Interval	Period	Action
No flexible intervals defined.		

New flexible interval

Interval (in sec)	50	Period	1-7,00:00-24:00	Add
-------------------	----	--------	-----------------	-----

History storage period (in days)

90

Trend storage period (in days)

365

Store value

As is

Show value

As is

show value mappings

New application

Applications

-None-

CPU

Filesystems

General

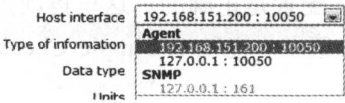

memory

Network interfaces

图 5-9

Items 的属性参数配置说明如表 5-1 所示。

表 5-1

选 项	说 明
Host	选择 Host 或者 Template，也就是说，Items 是存在于 Host 或者 Template 中的，不能单独存在
Name	Item 的名字，可以用宏（macros）变量：\$1, \$2, …, \$9，代表 Item 名称的第 1、2、…、9 参数。举例：Free disk space on \$1 如果 Item key 为“vfs.fs.size[/,free]”，那么名称将会变成“Free disk space on /”
Type	Item 的默认类型包括 Zabbix agent、Simple checks、SNMP、Zabbix internal、IPMI、JMX monitoring、Telnet、SSH 等多种监控方式。因监控方式的不同，Key 的配置也会不同
Key	Item 的 Key，通过选择菜单能查找出可以支持的 Item keys，但自定义的 Key 不能通过选择菜单查找出来，需要手动输入 对同一个主机来说，Key 必须是唯一的，不能有重复。如果 Key 类型为'Zabbix agent'、'Zabbix agent (active)'、'Simple check'或者'Zabbix aggregate'，Key 值必须能被 Zabbix-Agent 或 Zabbix-Server 所支持 对 Zabbix-Agent 所能支持的 Key，请读者查看官方文档 https://www.zabbix.com/documentation/2.2/manual/config/items/itemtypes/zabbix_agent
Host interface	选择主机的接口，在配置了多种监控方式或者多个 IP 监控的时候，可以基于主机接口来选择，如下图所示  其选项值来源于配置的多个监听接口，如下图所示 
Type of information	数据在进行类型转换之后存入数据库中 <ul style="list-style-type: none"> Numeric (unsigned): 64 位无符号的整数 Numeric (float): 浮点数类型（从 Zabbix 2.2 开始，接收的值大小可以支持如 1e+70、1e-70。） Character: 字符（字符串）类型数据限制为 255B Log: 日志文件，必须使用的 Key 为 log[] Text: 文本（text）不限制大小
Data type	数据类型用于存储 Items 中 Key 所获取的数值值，用于存储在不同的表中，如 history、history_str 等表 <ul style="list-style-type: none"> Boolean: 在数据存储的时候将原本的值替换为 0 或 1，TRUE 存储为 1，FALSE 存储为 0，所有的值都是区分大小写的。

续表

选 项	说 明
Data type	<p>TRUE: true、t、yes、y、on、up、running、enabled、available FALSE: false、f、no、n、off、down、unused、disabled、unavailable</p> <p>另外, 任何非 0 的数值都被认为是 TRUE, 0 被认为是 FALSE</p> <ul style="list-style-type: none"> • Octal: 八进制数的数值格式 • Decimal: 十进制数的数值格式 • Hexadecimal: 十六进制数的数值格式 <p>Zabbix 自动执行数据类型的格式转换</p>
Units	<p>如果设置了一个单位符号, Zabbix 将会处理接收到的数据, 并且把数值转换为需要显示的单位。默认情况下, 如果原始值超过 1000, 除以 1000, 并相应地显示。例如, 设置 bps 为单位, 如果接收到的值为 881764, 将会显示为 881.76Kbps</p> <p>特殊情况有: 当单位为 B (byte)、Bps (bytes 每秒) 时, 将会除以 1024, 如果单位设置为 B 或 Bps, Zabbix 将会显示: 1 为 1B/1Bps, 1024 为 1KB/1KBps, 1536 为 1.5KB/1.5KBps。</p> <p>特殊处理与时间相关的单位如下。</p> <ul style="list-style-type: none"> • Unixtime: 接收到的值转换为 “yyyy.mm.dd hh:mm:ss”。为了能正确显示, 接收到的值必须是一个数值类型 (unsigned)。 • Uptime: 接收到的值将会转换为 “hh:mm:ss” 或 “N days, hh:mm:ss”。例如, 接收的值为 881764 (seconds), 将会显示为 “10 days, 04:56:04”。 • S: 接收到的值将会转换为 “yy mmm dddhhh mmm sssms”, 参数为秒。 <p>例如, 如果接收到的值为 881764 (seconds), 将会显示为 “10d 4h 56m”。只显示 3 个主要的单位, 例如, “1m 15d 5h” 或 “2h 4m 46s”; 如果没有 day, 将只会显示 “1m 5h” (不会显示 minutes、seconds 或者 milliseconds); 如果值小于 0.001, 将会显示为 “< 1 ms”</p>
Use custom multiplier	<p>如果开启该选项, 所接收到的数值将会被乘以整数或者浮点数。用这个选项可以将 KB、Mbps 等转换为 B、Bps。另外, Zabbix 无法正确设置 prefixes (K、M、G 等单位) (从 Zabbix 2.2 开始, 支持科学计数法, 如 1e+70)</p>
Update interval (in sec)	<p>间隔时间 (秒) 通过 Item 收集数据。</p> <p>注意: 如果设置为 '0', Item 将不会通过 Key 采集数据, 如果一个灵活的间隔存在一个非零值, Item 将在灵活的间隔时间内轮询采集数据</p>
Flexible intervals	<p>创建一个非常规的更新时间 (如果同时设置了 Update interval 的值, 则以 Flexible intervals 设置的为准), 例如: Interval: 10, Period: 1-5, 09:00-18:00, 在工作时间, 每隔 10s 将会采集数据, 其他时间段内将使用默认的更新间隔时间 (update interval)。</p> <p>如果有多个更新时间间隔, 采集到的值将会是最小时间内的值。</p> <p>注意: 如果设置为 '0', Item 将不会在这个时间段内更新数据。Zabbix-Agent 在主动工作模式的 Items 是无效的</p>

续表

选 项	说 明
Keep history (in days)	<p>数据库中保存历史数据的天数，这个值对 Zabbix 数据库的大小有非常重要的影响，超过时间设置的数据，将会被 Housekeeper 功能进行清理</p> <p>从 Zabbix 2.2 开始，Housekeeper 的配置可以通过全局设置（单击 Administration→General→Housekeeper）。如果全局设置已经存在，将会有有一个警告信息显示出来：</p> <p>Keep history (in days) <input type="text" value="14"/> Overridden by global housekeeper settings (7 days)</p> <p>建议将历史记录值保存的最小天数尽可能地减少，如果要保存比较长的历史记录，一个替代的方法是将趋势数据（trends）增大</p>
Keep trends (in days)	<p>在数据库中保存历史的趋势数据（每小时的最小值、最大值、平均值、统计），超过时间设置的数据，将会被 Housekeeper 进程清理掉</p> <p>注意：trends 数据不能保存为非数值的数据，如字符、日志和文本</p>
Store value	<ul style="list-style-type: none"> As is: 无预处理 Delta (speed per second): 数值计算的方法为 $(\text{value} - \text{prev_value}) / (\text{time} - \text{prev_time})$，其中，各项的含义如下。 value: 当前的值； value_prev: 上一次收到的值； time: 当前时间； prev_time: 上一次接收到值的时间。 这个设置对于想显示每秒速率的值非常有用。 注意：如果当前的值小于上一次的值，Zabbix 将不记录此值（存储为空），会等待下一个值。这能使值的显示处于一个正常状态，例如，32-bit SNMP counters 会发生溢出的问题（从而导致断图），参考链接 https://www.zabbix.com/forum/showthread.php?t=10811 Delta (simple change): 数值计算的方法为 $(\text{value} - \text{prev_value})$，其中，value 表示当前值；value_prev 表示上一次接收到的值
Show value	<p>对 Item 的值做一个值的视图显示。Value mapping 不会改变接收到的值，仅仅改变显示，通过其他字符显示值的状态。仅支持整数类型的 Items，例如，“Windows service states”</p>
Log time format	<p>Items 类型为日志时，支持如下占位符。</p> <ul style="list-style-type: none"> y: Year (0001~9999) M: Month (01~12) d: Day (01~31) h: Hour (00~23) m: Minute (00~59) s: Second (00~59) <p>例如，Zabbix agent log 文件“23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211).”中，开头 6 个字符是 PID，接下来是日期、时间和该行其他的内容。</p> <p>Log 的时间格式将会是 “pppppp:yyyyMMdd:hhmmss”</p> <p>注意：这里的“p”和“:”只是一个占位符，除了“yMdhms”，可以为其他任意格式</p>

续表

选 项	说 明
New application	输入一个名称将会把 Item 加入一个新的 Application
Applications	将 Item 添加到一个或者多个已经存在的 Applications 中
Populates host inventory field	将 Item 归属于哪个资产管理的组中
Description	对 Item 的描述
Enabled	勾选 Enable，将开启 Item

5.2 Items key 的添加

1. Key 的格式

Key 可以带参数，该参数为一个数组列表，可以同时传递多个参数，Key 的格式如图 5-10 所示。

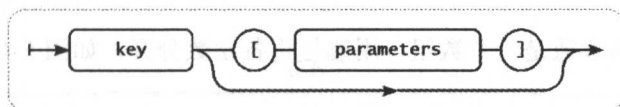


图 5-10

即 Key 的格式为 Key 或者是 Key[接参数]，例如

```
vfs.fs.size[/]
vfs.fs.size[/opt]
```

Key 的可接参数分为引号字符串、非引号字符串和数组，如图 5-11 所示。

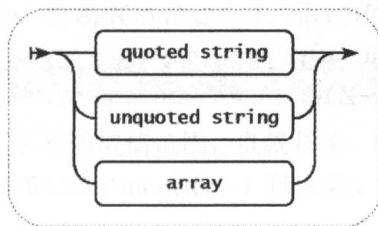


图 5-11

2. 引号字符串参数

如果参数是引号字符串，其中可以为任意字符串，但如果存在双引号，必须用反斜杠 (\) 进行转义，如图 5-12 所示。

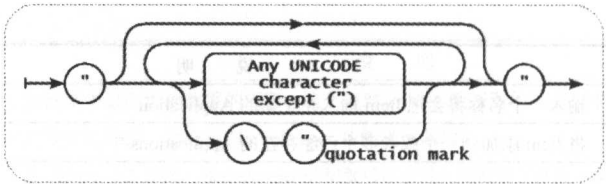


图 5-12

3. 非引号字符串参数

除逗号、右方括号外的其他字符都能引用，如图 5-13 所示。

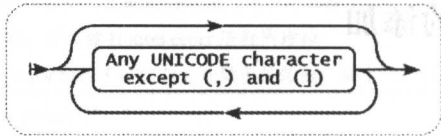


图 5-13

4. 数组

如果 Key 的参数是一个数组，用逗号将各参数分开，如图 5-14 所示。

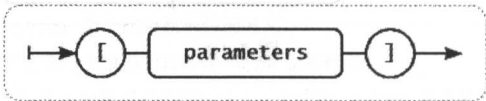


图 5-14

5. Key 的名称定义注意事项

Key 名称的取值范围如下。

- 所有的数字（0~9）；
- 所有的小写字母（a~z）；
- 所有大写字母（A~Z）；
- 下画线（_）；
- 破折号（——）；
- 点（.）。

6. Key 的参数数组应用实例

多个参数用逗号（,）分开，每个参数对 Key 分别传递参数的值。

```
UserParameter=wc[*],grep -c "$2" $1
```

上述语句表示把\$2、\$1 的值传递给 Key，测试如下。

```
shell# zabbix_get -s 127.0.0.1 -k wc[/etc/passwd,root]
```

注意,这里的/etc/passwd 为\$1,root 为\$2,则 Key 最终运行的命令为 `grep -c root /etc/passwd`。

如果方括号[]中有多个参数选项的值,每一个参数用逗号隔开,如:

```
icmpping[,200,,500]
```

7. 用户自定义参数

用户自定义参数 (UserParameter) 仅支持 Agent 的方式,对于其他方式,它是不支持的。

(1) Key 自定义的语法格式

在/etc/zabbix/zabbix_agentd.conf 中配置参数,写法如下。

```
UserParameter=key,command
```

除了上面这种写法,还支持参数传递的写法,具体如下。

```
UserParameter=key[*],command $1 $2 $3 .....
```

上述语句中,参数的含义如表 5-2 所示。

表 5-2

参 数	描 述
Key	Item key 具有唯一性,定义[*]可以接收参数
Command	Zabbix 将[]中的参数传递给命令中的\$1, ..., \$9, 将值作为命令的一部分

(2) 自定义 Key 中对特殊字符的处理

如果 UserParameter 包含' " `*?[]{}~\$!&;()<>|#@这些字符,默认情况下,Zabbix 对这些参数是无法正常处理的,需要在 zabbix_agentd.conf 中开启参数 UnsafeUserParameters,并将其值设置为 1,语句如下。

```
shell# vim /etc/zabbix/zabbix_agentd.conf
UnsafeUserParameters=1
```

前面已经介绍了 Key 名称的取值范围,也就是说,UserParameter 中的命令包含特殊字符必须开启 UnsafeUserParameters=1 的参数,然后重启 Zabbix_Agentd 服务。

默认情况下,\$1、\$2、\$3、\$4 代表位置参数 1、2、3、4,如果定义的字符串中出现\$后面接数字,需要用\$\$,示例如下。

```
awk '{print $$2}'
```

(3) Key 返回的值

自定义参数可以返回文本 (character、log、text) 和空值,如果返回的是一个无效值,则显示 ZBX_NOTSUPPORTED。

(4) 自定义 Key 的例子

在 /etc/zabbix/zabbix_agentd.conf 后面添加如下内容。

```
UserParameter=get.os.type, head -1 /etc/issue
```

然后重启 zabbix_agentd 服务（注意，修改配置后必须重启服务）。

```
shell# service zabbix_agentd restart
```

运行测试命令，查看 Key，语句如下。

```
shell# zabbix_get -s 127.0.0.1 -k get.os.type
CentOS release 6.5 (Final)
```

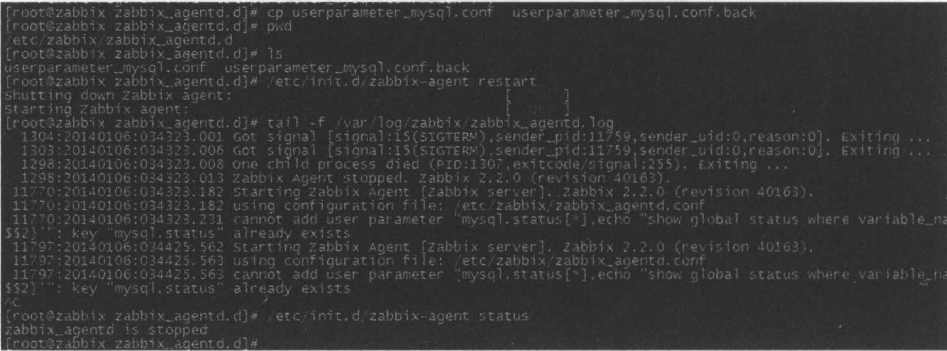
(5) 子配置文件的配置

为了便于维护和分类管理，UserParameter 的内容可以单独写一个配置文件。

```
shell# vim /etc/zabbix/zabbix_agentd.conf
Include=/etc/zabbix/zabbix_agentd.conf.d/
```

Include 也就是 zabbix_agent.conf 中部分内容的子配置文件，这种方式在其他开源软件中也是类似的做法，便于维护。子配置文件中通常会定义 UserParameter。

注意，在/etc/zabbix/zabbix_agentd.conf.d/ 文件夹下所有的配置文件都会被当作子配置文件，如果有重复的，会导致客户端 Agent 启动失败，如图 5-15 所示。



```
[root@zabbix zabbix_agentd.d]# cp userparameter_mysql.conf userparameter_mysql.conf.back
[root@zabbix zabbix_agentd.d]# pwd
/etc/zabbix/zabbix_agentd.d
[root@zabbix zabbix_agentd.d]# ls
userparameter_mysql.conf userparameter_mysql.conf.back
[root@zabbix zabbix_agentd.d]# /etc/init.d/zabbix-agent restart
Shutting down Zabbix agent:
Starting Zabbix agent:
[root@zabbix zabbix_agentd.d]# tail -f /var/log/zabbix/zabbix_agentd.log
1301:20140106:034323.001 got signal [signal:15(SIGTERM),sender_pid:11759,sender_uid:0,reason:0]. Exiting ...
1303:20140106:034323.006 got signal [signal:15(SIGTERM),sender_pid:11759,sender_uid:0,reason:0]. Exiting ...
1298:20140106:034323.008 one child process died (pid:1307,exitcode:signal(255)). Exiting ...
1298:20140106:034323.013 Zabbix Agent stopped. Zabbix 2.2.0 (revision 40163).
11770:20140106:034323.182 Starting Zabbix Agent [Zabbix server]. Zabbix 2.2.0 (revision 40163).
11770:20140106:034323.182 using configuration file: /etc/zabbix/zabbix_agentd.conf
11770:20140106:034323.231 cannot add user parameter 'mysql.status[*].echo' 'show global status where variable_name = *': key 'mysql.status' already exists
11797:20140106:034425.562 Starting Zabbix Agent [Zabbix server]. Zabbix 2.2.0 (revision 40163).
11797:20140106:034425.563 using configuration file: /etc/zabbix/zabbix_agentd.conf
11797:20140106:034425.563 cannot add user parameter 'mysql.status[*].echo' 'show global status where variable_name = *': key 'mysql.status' already exists
[root@zabbix zabbix_agentd.d]# /etc/init.d/zabbix-agent status
zabbix_agentd is stopped
[root@zabbix zabbix_agentd.d]#
```

图 5-15

(6) 用户自定义参数的总结

- 自定义 Key 的语法。
- 特殊字符的处理。
- 子配置文件的注意事项。
- 自定义 Key 的步骤总结。

1) Agent 配置文件修改。

```
shell# vim /etc/zabbix/zabbix_agentd.conf
UnsafeUserParameters=1
```

#处理特殊字符

```
Include=/etc/zabbix/zabbix_agentd.conf.d/ #子配置文件路径
```

2) 子配置文件。

```
shell# vim /etc/zabbix/zabbix_agentd.conf.d/get_os_type.conf
UserParameter=get.os.type, head -1 /etc/issue
#自定义key, 如有参数传递, 参考前面的内容
```

3) 重启服务测试 Key。

```
shell# service zabbix-agent restart #重启服务
shell# zabbix_get -s 127.0.0.1 -k get.os.type #测试Key获取参数
CentOS release 6.4 (Final) #Key获取的值
```

4) 用 zabbix_agentd 查看 Key 是否被支持。

```
shell# zabbix_agentd -p|grep get\.os
get.os.type [t|CentOS release 6.5 (Final)]
```

如果能看到 Key 名称, 且能看到获取到的数据, 说明自定义的 Key 是正确的。

5) 在 Web 页面添加 Item, 注意数据类型的选择。

5.3 Items 的类型

Zabbix 的 Items 分为多种类型, 这里只讨论主要的用法, 对不常用的, 读者可自行参考官方文档。

<https://www.zabbix.com/documentation/2.2/manual/config/items/itemtypes>

5.3.1 Zabbix-Agent

Agent 用于从 Zabbix-Agent 采集数据, 其工作方式分为被动模式 (Passive) 和主动模式 (Active), 在默认的模板中, 是被动模式的工作方式。

Zabbix-Agent 支持的 Item keys 请读者参考本书附录部分, 官方文档地址为:

https://www.zabbix.com/documentation/2.2/manual/config/items/itemtypes/zabbix_agent

下面对 Linux 下 Agent key 的各种用法进行说明, 并对常用的 Key 进行分类。

1. 网卡流量的监控 Key

```
net.if.in[if, <mode>]
```

其中, if 表示网卡接口, mode 表示想要取值的类型。通过查看文档可知, mode 的可选参数为 bytes (默认)、packets、errors、dropped。

注意, 凡是文档中所提的默认, 在多个参数的时候, 不填写参数, 用逗号分隔开, 即使用默认的参数。

例如, `net.if.in[if,<mode>]` 这个 Key, `mode` 默认的参数是 `bytes`, 所以 `net.if.in[eth0]` 表示获取 `eth0` 的流量 (`bytes`), 与 `net.if.in[eth0,bytes]` 效果相同。

用 `zabbix_get` 测试数据获取情况 (`zabbix_get` 的用法请参考 3.7 节)。

```
[root@www ~]# zabbix_get -s 127.0.0.1 -k net.if.in[eth0,bytes]
358589160
```

这里获取的值就是网卡的进流量。

如果想获取网卡接收的数据包数量, 用 `net.if.in[eth0,packets]` 即可。

```
[root@www ~]# zabbix_get -s 127.0.0.1 -k net.if.in[eth0,packets]
257021
```

这里的 `packets` 其实就是 `ifconfig` 命令中看到的 `packets`。

```
[root@www ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:12:F6:05
          inet addr:192.168.1.9 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe12:f605/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:257021 errors:0 dropped:0 overruns:0 frame:0
          TX packets:165997 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:357444550 (340.8 MiB)  TX bytes:11420540 (10.8 MiB)
```

`errors` 和 `dropped` 的原理相同, 在此不再重复。

另外, 关于网卡流量的 Key 如下 (具体用法见文档解释)。

```
net.if.out[if,<mode>]
net.if.collisions[if]
net.if.discovery
net.if.out[if,<mode>]
net.if.total[if,<mode>]
```

2. 监控端口的 Key

相关的 Key 如下。

```
net.tcp.listen[port]
net.tcp.port[<ip>,port]
net.tcp.service[service,<ip>,<port>]
net.tcp.service.perf[service,<ip>,<port>]
net.udp.listen[port]
```

3. 监控进程的 Key

```
kernel.maxfiles
kernel.maxproc
proc.mem[<name>,<user>,<mode>,<cmdline>]
proc.num[<name>,<user>,<state>,<cmdline>]
```

4. 监控 CPU 和内存的 Key

```
system.cpu.intr
```



```

system.cpu.load[<cpu>,<mode>]
system.cpu.num[<type>]
system.cpu.switches
system.cpu.util[<cpu>,<type>,<mode>]
vm.memory.size[<mode>]
system.swap.in[<device>,<type>]
system.swap.out[<device>,<type>]
system.swap.size[<device>,<type>]

```

5. 磁盘 I/O 的 Key

```

vfs.dev.read[<device>,<type>,<mode>]
vfs.dev.write[<device>,<type>,<mode>]
vfs.fs.inode[fs,<mode>]

```

6. 文件监控的 Key

```

vfs.file.cksum[file]
vfs.file.contents[file,<encoding>]
vfs.file.exists[file]
vfs.file.md5sum[file]
vfs.file.regexp[file,regexp,<encoding>,<start line>,<end line>,<output>]
vfs.file.regmatch[file,regexp,<encoding>,<start line>,<end line>]
vfs.file.size[file]
vfs.file.time[file,<mode>]
Vfs.fs.discovery
vfs.fs.size[fs,<mode>]

```

7. 日志监控的 Key

需要主动模式的支持

```

log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>]
logrt[file_pattern,<regexp>,<encoding>,<maxlines>,<mode>,<output>]

```

关于更多的 Key,读者可以参考本书的附录部分。官方文档地址为 https://www.zabbix.com/documentation/2.2/manual/config/items/itemtypes/zabbix_agent。

8. Windows 专用的 Key

```

eventlog[name,<regexp>,<severity>,<source>,<eventid>,<maxlines>,<mode>]
net.if.list
perf_counter[counter,<interval>]
proc_info[<process>,<attribute>,<type>]
service_state[*]
services[<type>,<state>,<exclude>]
wmi.get[<namespace>,<query>]

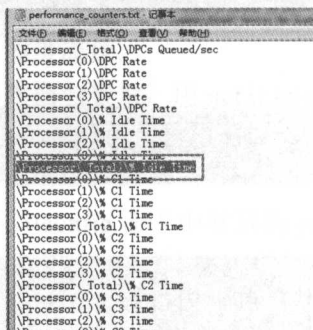
```

关于 Key 的详细用法,请读者参考如下地址。

https://www.zabbix.com/documentation/2.2/manual/config/items/itemtypes/zabbix_agent/win_keys

在上面的 Key 中, `perf_counter[counter,<interval>]` 可以用来获取 Windows 的性能监视器的参数, 用这个 Key 可以获取非常多的 Windows 性能参数值, 性能监视器可以用 `typeperf` 命令来查看参数。

C:\typeperf -qx > **performance_counters.txt** # 查看结果如图 5-16 所示



```

performance_counters.txt 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
\Processor(_Total)\DPCs Queued/sec
\Processor(0)\DPC Rate
\Processor(1)\DPC Rate
\Processor(2)\DPC Rate
\Processor(3)\DPC Rate
\Processor(_Total)\DPC Rate
\Processor(0)\% Idle Time
\Processor(1)\% Idle Time
\Processor(2)\% Idle Time
\Processor(3)\% Idle Time
\Processor(_Total)\% Idle Time
\Processor(0)\% C1 Time
\Processor(1)\% C1 Time
\Processor(2)\% C1 Time
\Processor(3)\% C1 Time
\Processor(_Total)\% C1 Time
\Processor(0)\% C2 Time
\Processor(1)\% C2 Time
\Processor(2)\% C2 Time
\Processor(3)\% C2 Time
\Processor(_Total)\% C2 Time
\Processor(0)\% C3 Time
\Processor(1)\% C3 Time
\Processor(2)\% C3 Time
\Processor(3)\% C3 Time
\Processor(_Total)\% C3 Time
\Processor(0)\% DPCs/sec
\Processor(1)\% DPCs/sec
\Processor(2)\% DPCs/sec
\Processor(3)\% DPCs/sec
\Processor(_Total)\% DPCs/sec

```

图 5-16

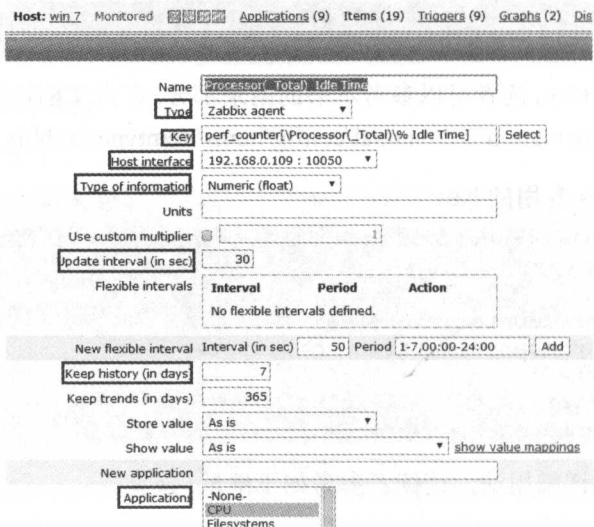
在 Windows 中安装好 Zabbix-Agent 后, 配置 `zabbix_agentd.conf`, 即可采集数据。

```

C:\>"C:\Program Files\zabbix_agents_2.2.0.win\in\win64\zabbix_agentd.exe" -c "c:\Program Files\zabbix_agents_2.2.0.win\conf\zabbix_agentd.win.conf" -t perf_counter["\Processor(_Total)\% Idle Time"]
perf_counter[\Processor(_Total)\% Idle Time] [d|92.035326]

```

添加 Key 到主机, 如图 5-17 所示。



Host: win_7 Monitored Applications (9) Items (19) Triggers (9) Graphs (2) Dis

Name:

Type:

Key:

Host interface:

Type of information:

Units:

Use custom multiplier: ☐

Update interval (in sec):

Flexible intervals

Interval	Period	Action
No flexible intervals defined.		

New flexible interval: Interval (in sec) Period Add

Keep history (in days):

Keep trends (in days):

Store value:

Show value: show value mappings

New application:

Application:

图 5-17

在 Windows 中需要更改 Windows 防火墙,如果是 Windows 7 和 Windows 2008,在开启防火墙的情况下,需开启入站规则 10050 端口才能被访问,如图 5-18 所示。

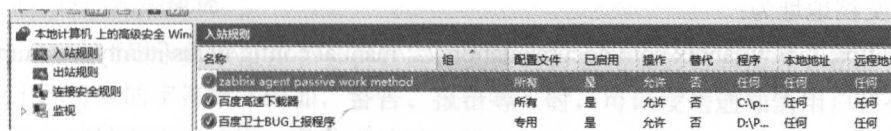


图 5-18

如果是主动模式,则在出站规则中增加 10051 端口的访问许可即可。

在 Zabbix-Server 服务器中测试能否获取到数据。

```
shell# zabbix_get -s 192.168.0.109 -k system.uname
Windows ITNIAHO 6.1.7601 Microsoft Windows 7 Ultimate Edition Serv
ice Pack 1 x64
```

在 Zabbix 的 Web 界面添加图形后,即可查看监控到的数据,如图 5-19 所示。

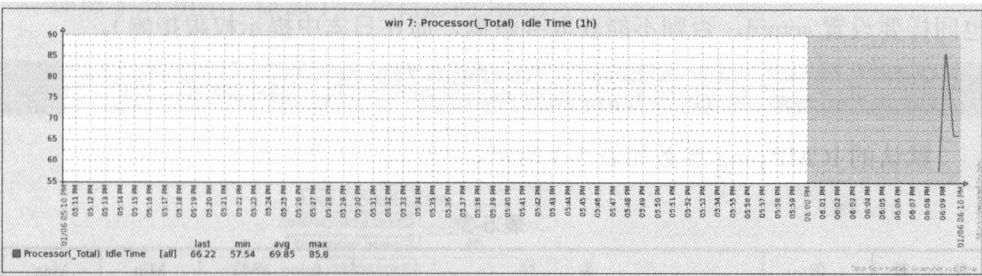


图 5-19

添加 Items 之后,按照需要添加 Graphs、Trigger 等。如果需要对其他参数进行监控,按照以上方法直接添加 Key 即可。

在某些特殊需求的情况下,可能需要对 Windows 的性能监视计数器进行自定义,可以参考 <http://support.microsoft.com/kb/317679>。

5.3.2 Simple check

Simple check 用于无须远程客户端的环境,即没有使用 Agent 的情况下,其可支持的功能包括 icmp ping 和 service 检测。除此之外,Simple check 在 Zabbix 2.2 以后支持 VMware 的监控,这部分内容请读者参考监控 VMware 的内容。

1. Simple check 支持的 Key

```
icmpping[<target>,<packets>,<interval>,<size>,<timeout>]
icmppingloss[<target>,<packets>,<interval>,<size>,<timeout>]
icmppingsec[<target>,<packets>,<interval>,<size>,<timeout>,<mode>]
net.tcp.service[service,<ip>,<port>]
```

```
net.tcp.service.perf[service,<ip>,<port>]
```

关于 Simple check 所支持的 Key 的详细内容,读者可以参考本书的附录部分,官方文档地址为:

https://www.zabbix.com/documentation/2.2/manual/config/items/itemtypes/simple_checks

2. 超时处理

超过 zabbix_server.conf 中设置的超时时间范围后, Zabbix 会放弃处理。

3. ICMP ping

Zabbix 用 fping 处理 ICMP ping 请求,所以需要安装 fping 程序,在 zabbix_server.conf 中有一个参数 FpingLocation 是配置 fping 程序路径的。

由于 fping 默认是 root 权限工作,而 Zabbix-Server 是 Zabbix 用户运行的,所以需要对 fping 程序设置 setuid (如果在自定义 Key 的时候需要用到 netstat 命令,也同样要设置 setuid,否则不能获取到数据,而在日志中提示权限拒绝)。

```
shell# chown root:zabbix /usr/sbin/fping
shell# chmod 4710 /usr/sbin/fping
```

默认的 ICMP ping 参数如表 5-3 所示。

表 5-3

参 数	值	备 注	fping 参数	Min	Max
packets	3	Ping 的次数	-C	1	10000
interval	1000	毫秒 (ms)	-p	20	
size	56 或 68	Bytes, x86 平台是 56B, x86_64 平台是 68B	-b	24	65507
timeout	500	毫秒 (ms)	-t	50	

示例如图 5-20 所示,其中的 icmpingloss[114.114.114.114,5]为 Key。

Item configuration form:

- Name: 114 ping 丢包率
- Type: Simple check
- Key: icmpingloss[114.114.114.114,5]
- User name:
- Password:
- Type of information: Numeric (unsigned)
- Data type: Decimal
- Units:
- Use custom multiplier: ☒ 1
- Update interval (in sec): 30
- Flexible intervals:

Interval	Period	Action
50	1-7,00:00-24:00	Remove

图 5-20

5.3.3 日志监控方式

1. 日志监控概述

Zabbix 可用于集中监控和分析日志，支持有日志轮询的日志监控分析。当日志中出现特殊的字符串（例如，警告、报错等）时，可以发送通知给用户。为了使日志监控能够正常使用，必须满足以下两个条件。

- Zabbix-Agent 必须运行，且工作模式为主动模式。
- 日志的 Items 必须设置。

注意：Zabbix 日志监控必须工作于主动模式，在 Web 前端配置的主机名必须和 Agent 端 zabbix_agentd.conf 中的 Hostname 值是一致的，并且这个 Hostname 具有唯一性，否则，主动模式是无法正常采集到数据的。

2. 日志监控 Items 的配置

如图 5-21 所示，添加了一个日志的 Items。

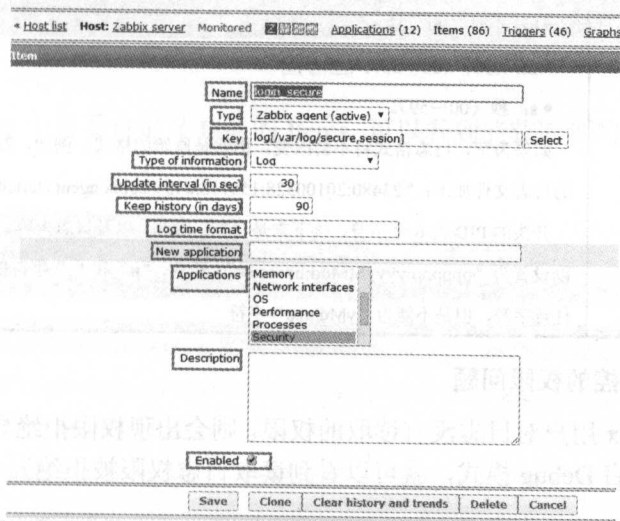


图 5-21

3. 日志监控的 Items 属性

日志监控的 Items 属性说明如表 5-4 所示。

表 5-4

Type	选择 Zabbix-Agent (active)监控方式
Key	有以下两个 Key: log[/path/to/file/file_name,<regexp>,<encoding>,<maxlines>,<mode>,<output>]

续表

Type	选择 Zabbix-Agent (active) 监控方式
Key	<p>或</p> <p><code>logrt[/path/to/file/regexp_describing_filename_pattern,<regexp>,<encoding>,<maxlines>,<mode>,<output>]</code></p> <p>Zabbix-Agent 可以对日志内容通过正则表达式过滤, 需要注意的是日志, Zabbix 用户必须对这个日志文件有读取的权限, 否则会提示 'unsupported'</p>
Type of information	选择 Log
Update interval (in sec)	该参数告诉 Zabbix-Agent 多长时间检测日志文件, 如果设置为 1 秒, 将会快速收集到日志, 但是会增加 Zabbix-Server 的负担
Log time format	<p>支持的符号如下。</p> <ul style="list-style-type: none"> • y: 年 (0001~9999); • M: 月 (01~12); • d: 日 (01~31); • h: 小时 (00~23); • m: 分钟 (00~59); • s: 秒 (00~59)。 <p>如果为空, 日志格式将不会改变, 仍然是原始的格式。例如, Zabbix-Agent 采集到的日志文件如下: "23480:20100328:154718.045 Zabbix agent started. Zabbix 2.2.2."</p> <p>开头的 PID 占 6 个字符, 接下来是日期、时间, 以及日志内容。这里的日志格式可以设置为 "pppppp:yyyyMMdd:hhmmss"。注意, "p" 和 ":" 字符仅为占位符, 可以为任意字符, 但是不能为 "yMdhms" 字符</p>

4. 日志监控的权限问题

如果 Zabbix 用户对日志没有读取的权限, 则会出现权限拒绝导致数据获取失败 (Agentd 开启 Debug 模式, 就可以看到读取日志权限被拒绝)。

```
[root@zabbix ~]# tail -f /var/log/zabbix/zabbix_agentd.log|grep log
                "key":"log[/var/log/secure,session,,50]",
22851:20140106:225042.451 In add_check() key:'log[/var/log/secur
e,session,,50]' refresh:30 lastlogsize:0 mtime:0
22851:20140106:225042.451 In process_log() filename:'/var/log/se
cure' lastlogsize:0
22851:20140106:225042.452 cannot open '/var/log/secure': [13] Pe
rmission denied
22851:20140106:225042.452 active check "log[/var/log/secure,sess
ion,,50]" is not supported
```

```
22851:20140106:225042.452 In process_value() key:'Zabbix server:
log[/var/log/secure,session,,50]' value:'(null)'
"key": "log[/var/log/secure,session,,50]",
```

解决这个问题时，需要设置/var/log/secure 的正确权限。

```
shell# chown zabbix.root /var/log/secure
```

Log 的 Items 正常的日志如图 5-22 所示。

```
22852:20140106:230512.676 In process_log() filename: 'var/log/secure' lastlogsize:9376
22851:20140106:230512.707 In process_log() filename: 'var/log/secure' lastlogsize:9376
22851:20140106:230542.773 In process_log() filename: 'var/log/secure' lastlogsize:9376
22852:20140106:230542.773 In process_log() filename: 'var/log/secure' lastlogsize:9376
22851:20140106:230612.822 In process_log() filename: 'var/log/secure' lastlogsize:9376
22852:20140106:230612.859 In process_log() filename: 'var/log/secure' lastlogsize:9376
"key": "log[/var/log/secure,session]",
22851:20140106:230642.876 In add_check() key: 'log[/var/log/secure,session]' refresh:30 lastlogsize:9376 mtime:0
22851:20140106:230642.877 In process_log() filename: 'var/log/secure' lastlogsize:9376
"key": "log[/var/log/secure,session]",
22852:20140106:230642.918 In add_check() key: 'log[/var/log/secure,session]' refresh:30 lastlogsize:9376 mtime:0
22852:20140106:230642.919 In process_log() filename: 'var/log/secure' lastlogsize:9376
22851:20140106:230712.959 In process_log() filename: 'var/log/secure' lastlogsize:9376
22851:20140106:230712.968 In process_log() filename: 'var/log/secure' lastlogsize:9376
22851:20140106:230742.004 In process_log() filename: 'var/log/secure' lastlogsize:9376
22852:20140106:230742.000 In process_log() filename: 'var/log/secure' lastlogsize:9376
22851:20140106:230812.081 In process_log() filename: 'var/log/secure' lastlogsize:9376
22852:20140106:230813.032 In process_log() filename: 'var/log/secure' lastlogsize:9376
"key": "log[/var/log/secure,session]",
22852:20140106:230842.097 In add_check() key: 'log[/var/log/secure,session]' refresh:30 lastlogsize:9376 mtime:0
"key": "log[/var/log/secure,session]",
22851:20140106:230842.128 In add_check() key: 'log[/var/log/secure,session]' refresh:30 lastlogsize:9376 mtime:0
22851:20140106:230842.129 In process_log() filename: 'var/log/secure' lastlogsize:9376
22852:20140106:230843.102 In process_log() filename: 'var/log/secure' lastlogsize:9376
```

图 5-22

在 Last data 中查看自定义 Log 的 Items，可以看到能获取到的数据，如图 5-23 所示。

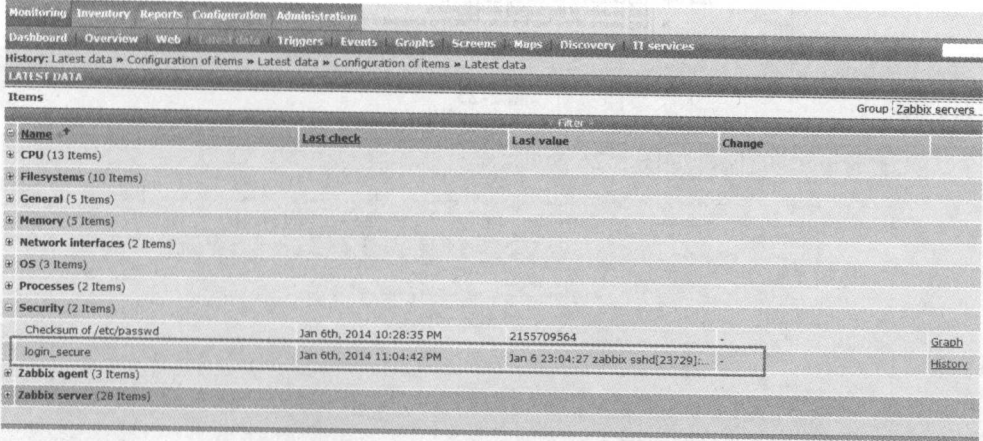


图 5-23

搜索过滤一下，只查看 Items 的 login_secure 数据，如图 5-24 所示。
当日志中符合定义的格式后，将会有数据从 Agent 发送到 Server。

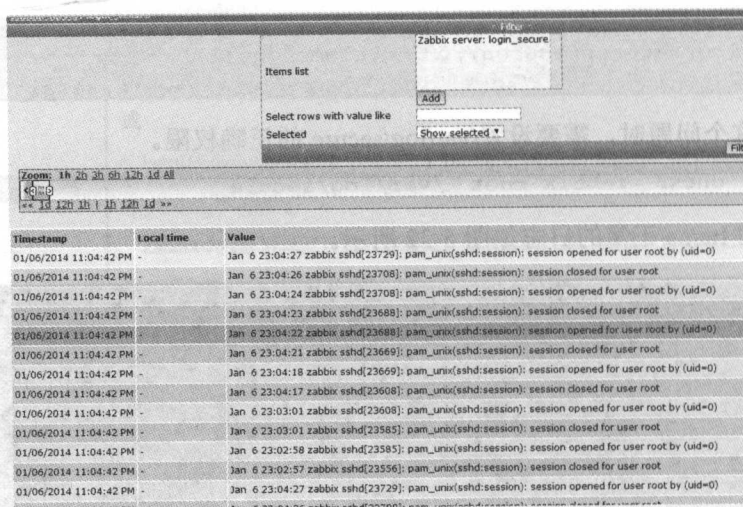


图 5-24

5. 日志监控触发器的设置

选择日志的 Item，如图 5-25 所示。

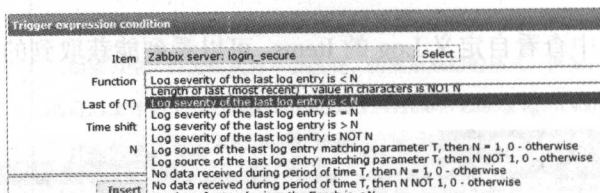


图 5-25

配置 Trigger 的值，如图 5-26 所示。

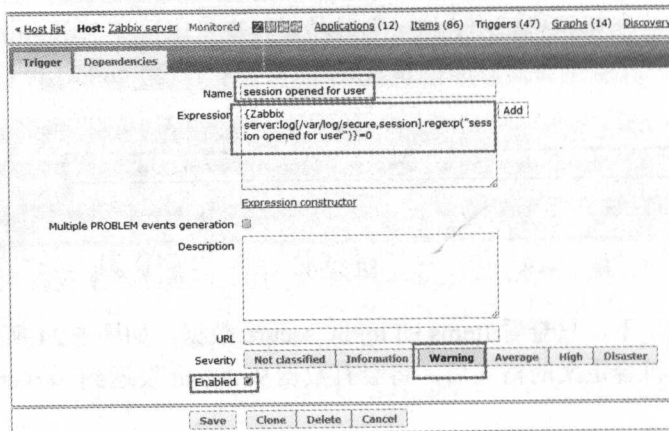


图 5-26

如果日志中出现“session opened for user”字符串，将会触发 Trigger，从而发送告警，如图 5-27 所示。

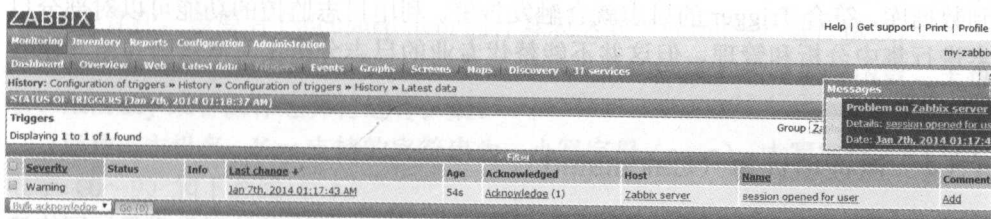


图 5-27

关于更复杂的 Trigger 设置，读者可以参考第 6 章关于 Trigger 的内容。

6. 日志监控的数据库

数据存储在 history_log 表中，表结构如图 5-28 所示。

```
mysql> desc history_log;
```

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	PRI	NULL	
itemid	bigint(20) unsigned	NO	MUL	NULL	
clock	int(11)	NO		0	
timestamp	int(11)	NO		0	
source	varchar(64)	NO		0	
severity	int(11)	NO		0	
value	text	NO		NULL	
logeventid	int(11)	NO		0	
ns	int(11)	NO		0	

图 5-28

```
mysql> select * from history_log; #如图5-29所示，是日志监控在数据库中的存储
```

```
mysql> select * from history_log;
```

id	itemid	clock	timestamp	source	severity	value
1	24696	1389020682	0	0 Jan 6 02:32:36 zabbix sshd[2075]: pam_unix(sshd:session): session closed for user root	0	
2	24696	1389020682	0	0 Jan 6 02:32:36 zabbix sshd[2075]: pam_unix(sshd:session): session opened for user root by (uid=0)	0	
3	24696	1389020682	0	0 Jan 6 22:46:27 zabbix sshd[22642]: pam_unix(sshd:session): session opened for user root by (uid=0)	0	
4	24696	1389020682	0	0 Jan 6 22:46:33 zabbix sshd[22642]: pam_unix(sshd:session): session closed for user root	0	
5	24696	1389020682	0	0 Jan 6 22:46:34 zabbix sshd[22642]: pam_unix(sshd:session): session opened for user root by (uid=0)	0	
6	24696	1389020682	0	0 Jan 6 22:46:35 zabbix sshd[22642]: pam_unix(sshd:session): session closed for user root	0	
7	24696	1389020682	0	0 Jan 6 22:46:36 zabbix sshd[22642]: pam_unix(sshd:session): session opened for user root by (uid=0)	0	
8	24696	1389020682	0	0 Jan 6 22:46:40 zabbix sshd[22642]: pam_unix(sshd:session): session closed for user root	0	
9	24696	1389020682	0	0 Jan 6 22:46:44 zabbix sshd[22642]: pam_unix(sshd:session): session opened for user root by (uid=0)	0	
10	24696	1389020682	0	0 Jan 6 22:46:48 zabbix sshd[22642]: pam_unix(sshd:session): session closed for user root	0	
11	24696	1389020682	0	0 Jan 6 22:46:48 zabbix sshd[22642]: pam_unix(sshd:session): session opened for user root by (uid=0)	0	
12	24696	1389020682	0	0 Jan 6 22:46:48 zabbix sshd[22642]: pam_unix(sshd:session): session closed for user root	0	
13	24696	1389020682	0	0 Jan 6 22:46:47 zabbix sshd[22642]: pam_unix(sshd:session): session opened for user root by (uid=0)	0	
14	24696	1389020682	0	0 Jan 6 22:46:51 zabbix sshd[22642]: pam_unix(sshd:session): session closed for user root	0	

图 5-29

需要注意的是，日志的工作方式必须为主动模式，用 zabbix_get 则提示不支持。

```
shell# zabbix_get -s 127.0.0.1 -k log[/var/log/secure,session]
```

ZBX NOTSUPPORTED

只有当日志发生改变，符合过滤条件时，才会发送给 Zabbix-Server，并记录到数据库，符合 Trigger 的日志就会触发告警。利用日志监控的功能可以对部分日志进行集中分析和管理工作，但这并不能替代专业的日志分析工具来解决日志集中管理工作。

5.3.4 监控项计算（Calculated）

用 Calculated items 可以对 Items 进行计算，例如，求总的磁盘容量、网络流量，计算主要是靠一系列的表达式组成的，只依赖于 Zabbix-Server，与 Zabbix-Agent 或者 Proxy 无关。

计算后的结果是存放在数据库中的，也就是说，历史数据、趋势数据都会保存在数据库中，Calculated items 可以用于 Trigger，这意味着可以对 Calculated items 进行告警配置，也可以被宏（macros）引用，其他配置与 Items 类型相同。

在 Calculated items 配置中，Key 和 Formula 是关键，如图 5-30 所示。

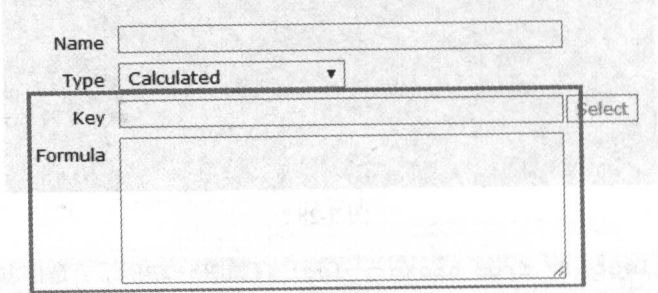


图 5-30

Key 在每个 Host 中必须是唯一的，不能重复。Formula 是计算表达式，其写法如下。

```
func(<key>|<hostname:key>,<parameter1>,<parameter2>,...)
```

上述语句中，各参数的含义如表 5-5 所示。

表 5-5

参数	含 义
func	支持 Trigger 正则表达式的函数：last、min、max、avg、count 等
key	key 可以被其他 Item 引用，可以被定义为 key 或者 hostname:key。 注意：建议将 key 放到 ("key") 双引号中，以避免不正确的解析（例如有空格或者分号的 key）。如果 key 中含有双引号，应该用斜杠 (\) 进行转义
parameter(s)	所需要的参数

1. 注意事项

所有被 Calculated item formula 引用的 Items 必须存在，且能收集数据，如果 Items 不存在，需要对 Calculated item 的计算表达式 (Formula) 进行修改。

在计算表达式中引用宏，可以对参数和常量进行扩展。但不能对函数、主机名、Item key 或者操作运算符进行扩展。

函数、主机名、Key 支持的字符串为：小写字母 (a~z)、大写字母 (A~Z)、数字 (0~9) 和下划线 (_).

不同于触发器的表达式，Zabbix 对 Items 的计算是根据 Items 的更新间隔周期进行数据更新的，不像 Items 必须等到接收到新的数据才对 Items 的值进行更新。

在以下情况中，不支持 Items 的计算。

- 引用的 Items 不被支持。
- 没有可用的数据进行计算操作。
- 除以 0。
- 不正确的语法操作。

2. 计算表达式 (Formula) 的例子

① 剩余磁盘的百分比。

```
100*last("vfs.fs.size[/,free]",0)/last("vfs.fs.size[/,total]",0)
```

② 10 分钟内 Zabbix values 的可用大小。

```
avg("Zabbix server:zabbix[wcache,values]",600)
```

③ 统计 eth0 的进出流量总和。

```
last("net.if.in[eth0,bytes]",0)+last("net.if.out[eth0,bytes]",0)
```

④ 统计进流量占网卡总流量的百分比。

```
100*last("net.if.in[eth0,bytes]",0)/(last("net.if.in[eth0,bytes]",0)+last("net.if.out[eth0,bytes]",0))
```

⑤ 对 Aggregated items 进行计算，注意引号需要转义。

```
last("grpsum[\"video\\\", \"net.if.out[eth0,bytes]\", \"last\\\", \"0\\\"]",0) / last("grpsum[\"video\\\", \"nginx_stat.sh[active]\", \"last\\\", \"0\\\"]",0)
```

3. 示例 1: 计算磁盘总的剩余容量

例如，要计算 C、D、E、F 磁盘总的剩余容量，分区剩余容量的 Key 为：

```
vfs.fs.size[C:,free]
```

```
vfs.fs.size[D:,free]
vfs.fs.size[E:,free]
vfs.fs.size[F:,free]
```

添加磁盘剩余总容量的计算表达式如图 5-31 所示。

```
last("vfs.fs.size[C:,free]",0)+last("vfs.fs.size[D:,free]",0)+last("vfs.fs.size[E:,free]",0)+last("vfs.fs.size[F:,free]",0)
```

« Host list Host: win 7 Monitored ☒ ☒ ☒ Applications (9) Items (98) Triggers (13) Graphs (37) Discovery rules (

Item

Name

Type

Key

Formula

Type of information

Units

Use custom multiplier ☐

Update interval (in sec)

Flexible intervals

Interval	Period	Action
No flexible intervals defined.		

New flexible interval Interval (in sec) Period

Keep history (in days)

Keep trends (in days)

Store value

Show value

New application

Applications ☒ -None- ☐ CPU ☒ Filesystems ☐ General

图 5-31

采集到的数据如图 5-32 所示。

Monitoring Inventory Reports Configuration Administration

Dashboard Overview Web Latest data Triggers Events Graphs Screens Maps Discovery IT services

History: History » Configuration of items » Configuration of templates » Configuration of items » Dashboard

LATEST DATA

Items

Host	Name ↑	Last check	Last value	Change
win 7	CPU (4 Items)			
win 7	Filesystems (22 Items)			
	Average disk read queue length	Jan 7th, 2014 10:39:03 AM	0.0026	-
	Average disk write queue length	Jan 7th, 2014 10:39:04 AM	0.01	-
	File read bytes per second	Jan 7th, 2014 10:39:05 AM	9.73 KBps	-795.25 Bps
	File write bytes per second	Jan 7th, 2014 10:39:06 AM	1.54 MBps	+1.33 MBps
	free disk	Jan 7th, 2014 10:39:07 AM	119812550656	-
	Free disk space on C:	Jan 7th, 2014 10:39:20 AM	3.48 GB	+60 KB
	Free disk space on C: (percentage)	Jan 7th, 2014 10:39:24 AM	7.13 %	-

图 5-32

单击 Graph，就可以看到 Simple graph 数据，如图 5-33 所示。

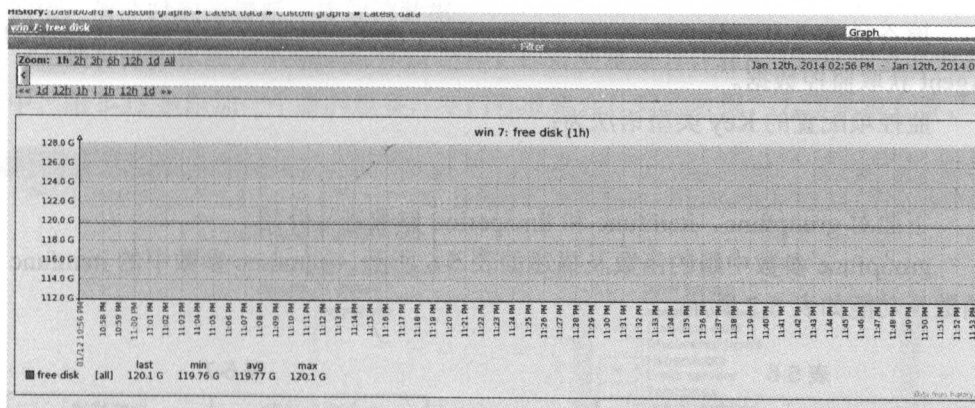


图 5-33

4. 示例 2：计算网卡的流量

表达式如图 5-34 所示。

```
last("net.if.in[eth0]",0)+last("net.if.in[eth1]",0)+last("net.if.in[eth2]",0)
```

Host list Host: Zabbix server Monitored Applications (12) Items (92) Triggers (47) Gra

Name net if in total on linux host

Type Calculated

Key net_if_in_total

Formula last("net.if.in[eth0]",0)+last("net.if.in[eth1]",0)+last("net.if.in[eth2]",0)

Type of information Numeric (float)

Units bps

Use custom multiplier 1

Update interval (in sec) 30

Flexible intervals

Interval	Period	Action
No flexible intervals defined.		

New flexible interval Interval (in sec) 50 Period 1-7,00:00-24:00 Add

Keep history (in days) 90

Keep trends (in days) 365

Store value As is

Show value As is show value mappings

图 5-34

5.3.5 聚合检测（Aggregate）

聚合检测是从数据库中对已经获取到的参数进行检测，它不需要从任何的 Agent 获取监控数据。

监控项配置的 Key 类型语法为：

```
groupfunc["Host group","Item key",itemfunc,timeperiod]
```

下面对 groupfunc、itemfunc 和 timeperiod 参数进行介绍。

groupfunc 参数中组的函数及描述如表 5-6 所示。itemfunc 参数中的 itemfunc 函数及功能如表 5-7 所示。

表 5-6

groupfunc 组的函数	描 述
grpavg	平均值
grpmax	最大值
grpmin	最小值
grpsum	值的个数

表 5-7

itemfunc 函数	功能描述
avg	平均值
count	值的个数
last	最新的值
max	最大值
min	最小值
sum	求和

timeperiod 参数是最近获取的数值，支持设置参数单位，例如，5m 和 300 是等价的，1d 和 86400 是等价的（时间单位中，不带单位时默认为秒）。如果 item 函数的参数为 last，timeperiod 参数将会被忽略。同时，last 函数中的#参数是不支持的（#参数代表最后第几次获取的数值）。

用法举例如下。

```
grpsum["MySQL Servers","vfs.fs.size[/,total]",last,0]
```

含义：对 MySQL Servers 组的 vfs.fs.size[/,total] key 计算个数，值的时间为最后一次取值数据。

```
grpavg["MySQL Servers","system.cpu.load[,avg1]",last,0]
```

含义：对 MySQL Servers 组的 system.cpu.load[,avg1]求平均值，值的时间为最后一次取值数据。

```
grpavg["MySQL Servers",mysql.qps,avg,5m]
```

含义：对 MySQL Servers 组中的各个监控项 mysql.qps 的平均值计算总体的平均值，值的时间为最近 5 分钟的数据。

```
grpavg[["Servers A","Servers B","Servers C"],system.cpu.load,last,0]
```

含义：对 Servers A、Servers B、Servers C 三个主机组的 system.cpu.load 求平均值，值的时间为最后一次取值数据。

聚合计算存在于 Template/Host 之上，即需要将聚合计算的 Items 建立在 Host 中，图 5-35 是添加一个新的设备，专门存放聚合计算的数据，方便单独查看。

CONFIGURATION OF HOSTS

Host Templates IPMI Macros Host inventory

Host name: Aggregate data

Visible name: Aggregate

Groups: In groups: Aggregate

Other groups: cloudstack-manager, Cplic-kvm-Host, Discovered hosts, Hypervisors, Linux servers, Templates, Virtual machines, Zabbix servers

New group:

Agent interfaces: IP address: 127.0.0.1, DNS name: , Connect to: IP, Port: 10050

SNMP interfaces: Add

JMX interfaces: Add

IPMI interfaces: Add

Monitored by proxy: (no proxy)

Status: Monitored

Save Cancel

图 5-35

监控方式必须选择其中一种，并填入相应的数据，否则页面会提示未输入数据。然后添加 Items 即可，前面已经讲解了如何添加 Items，此处不再重复。

实例：对 kvm-host 主机组的/分区剩余大小求和。Key 的写法如下。

```
grpsum["kvm-Host","vfs.fs.size[/,free]",last,0]
```

配置如图 5-36 所示。配置好之后，在间隔的时间结束后，即可获取数据，如图 5-37 所示。

对 “kvm-Host”、“Discovered hosts” 这两个主机组的/分区剩余大小求和，如图 5-38 所示。

```
grpsum[["kvm-Host","Discovered hosts"],"vfs.fs.size[/,free]",last,0]
```

CONFIGURATION OF ITEMS

Host list

Host: Aggregate

Monitored

Applications (0)

Items (2)

Triggers (0)

Graphs (0)

Item

Name

kvm-host group disk / free

Type

Zabbix aggregate

Key

grpsum["kvm-host","vfs.fs.size[/,free]",last,0]

Select

Type of information

Numeric (float)

Units

Use custom multiplier

☐

1

Update interval (in sec)

30

Flexible intervals

Interval	Period	Action
No flexible intervals defined.		

New flexible interval

Interval (in sec)	50	Period	1-7,00:00-24:00	A
-------------------	----	--------	-----------------	---

Keep history (in days)

90

Keep trends (in days)

365

Store value

As is

Show value

As is

show value map

New application

图 5-36

Monitoring Inventory Reports Configuration Administration

Dashboard Overview Web Latest data Triggers Events Graphs Screens Maps Discovery IT services

History: Configuration of items » Latest data » Configuration of items » Latest data » Configuration of items

LATEST DATA

Items

Name *	Last check	Last value
- other - (2 Items)		
kvm-host group disk / free	18 Feb 2014 17:56:34	895623495680

图 5-37

Host list Host: Aggregate Monitored Applications (0) Items (2) Triggers (0) Graphs (0) Discover

Item

Name

two groups disk / free space

Type

Zabbix aggregate

Key

grpsum[["kvm-host","Discovered hosts"],"vfs.fs.size[/,free]",last,0]

Type of information

Numeric (float)

Units

Use custom multiplier

☐

1

Update interval (in sec)

5

Flexible intervals

Interval	Period	Action
No flexible intervals defined.		

New flexible interval

Interval (in sec)	50	Period	1-7,00:00-24:00	Add
-------------------	----	--------	-----------------	-----

Keep history (in days)

90

Keep trends (in days)

365

Store value

As is

Show value

As is

show value mappings

New application

Applications

None

图 5-38

在 Latest data 中查看数据，如图 5-39 所示。

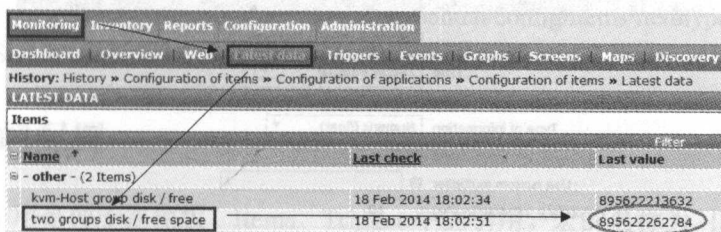


图 5-39

单击图 5-40 右边的 Graphs，可以看到图形数据，如图 5-40 所示。

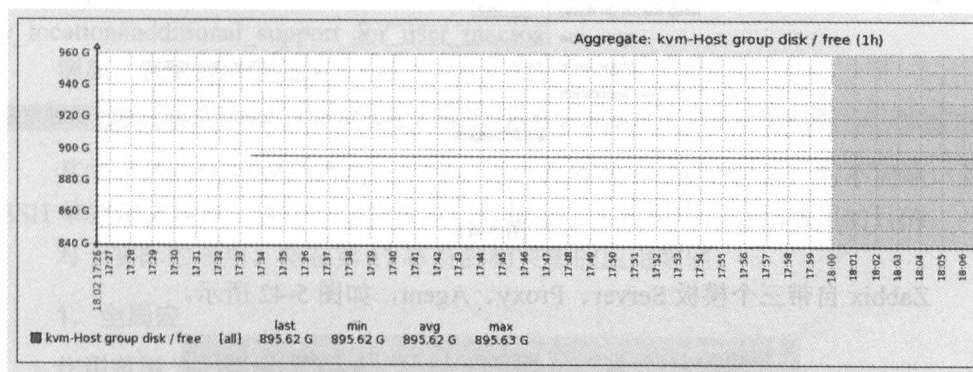


图 5-40

扩展：在现实环境中，我们可能需要对多个主机中多个磁盘分区的剩余空间进行数据汇总，其思路如下。

① 在各主机中用 Zabbix 计算的方式对各个磁盘分区求和，形成一个新的 Items(total free)。

② 用聚合计算对主机组的 Items (total free) 求和。

5.3.6 内部检测 (Internal)

内部检测用于监控 Zabbix 自身的性能数据，可监控 Zabbix-Server 或 Zabbix-Proxy，选择监控方式为 Zabbix internal，添加相应的 Item key，即可完成监控。

内部检测的数据格式可以为 zabbix[参数, 模式]。对于所有的 Key，读者可以参考如下地址的内容。

<https://www.zabbix.com/documentation/2.2/manual/config/items/itemtypes/internal>
Items 的添加方式如图 5-41 所示。

« Template list **Template: Template App Zabbix Server** Applications (1) Items (31) Triggers (27) Graphs (5)

Item

NameZabbix \$2 write cache, % free

TypeZabbix internal

Keyzabbix{wcache_history,pfree}Select

Type of informationNumeric (float)

Units%

Use custom multiplier☐ 1

Update interval (in sec)60

Flexible intervals

Interval	Period	Action
No flexible intervals defined.		

New flexible intervalInterval (in sec)50Period1-7,00:00-24:00Add

Keep history (in days)7

Keep trends (in days)365

Store valueAs is

Show valueAs isshow value mapping

New application

Applications

-None-Zabbix server

图 5-41

Zabbix 自带三个模板 Server、Proxy、Agent，如图 5-42 所示。

<input type="checkbox"/> Template App Zabbix Agent	Applications (1)	Items (3)	Triggers (3)	Graphs (0)
<input type="checkbox"/> Template App Zabbix Proxy	Applications (1)	Items (21)	Triggers (19)	Graphs (4)
<input type="checkbox"/> Template App Zabbix Server	Applications (1)	Items (31)	Triggers (27)	Graphs (5)

图 5-42

读者可根据实际需要 对 Items 进行增减。

5.3.7 SSH、Telnet 和扩展检测

对于 SSH 监控方式，其思路是利用 SSH 公钥，Zabbix Server 和被监控端无密码访问执行 SSH 命令，获取相关的 Shell 命令返回值。

读者如有需要，请参考官方文档，地址如下。

https://www.zabbix.com/documentation/2.2/manual/config/items/itemtypes/ssh_checks
进行 Telnet 检测时需要用户名和密码，Telnet 验证成功后，执行命令，获取返回值。官方文档地址为：

https://www.zabbix.com/documentation/2.2/manual/config/items/itemtypes/telnet_checks

扩展检测是通过带参数的脚本执行命令返回结果的。

官方文档地址为：

<https://www.zabbix.com/documentation/2.2/manual/config/items/itemtypes/external>
有关 SNMP JMX 方式的内容，将在第 7 章中讲解。

5.4 宏的配置

宏的作用是便于在模板、Items、Trigger 中的引用。宏的名称为 {\$名称}，宏的字符范围为 A~Z、0~9、_。

Zabbix 自带宏的官方参考文档中介绍了详细的内容，网址如下。

https://www.zabbix.com/documentation/2.2/manual/appendix/macros/supported_by_location#additional_support_for_user_macros

例如，在 Key 中的宏：

```
net.tcp.service[ssh,{SSH_PORT}]
```

其中，{\$SSH_PORT}就是一个宏，可以在添加 Items 的时候，对不同端口的 SSH 单独定义端口，这样一个模板就可以被多个主机引用，达到通用的目的。

对于实际的应用，将在第 11 章看到相关宏的应用案例。

1. 全局宏

作用范围为：模板、主机。

配置步骤为：单击 Administration→General→Macros，如图 5-43 所示。

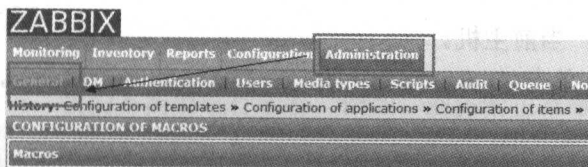
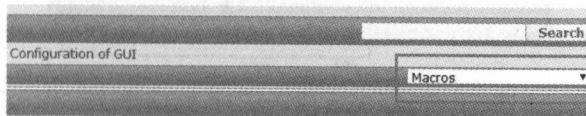


图 5-43

选择 Macros 下拉菜单，如图 5-44 所示。



Remove

图 5-44

默认的宏为{\$SNMP_COMMUNITY}，如图 5-45 所示。

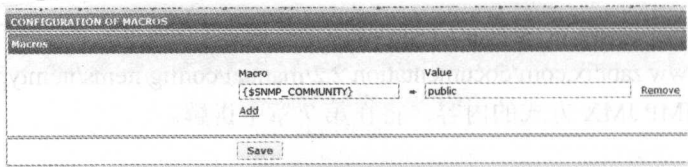


图 5-45

2. 模板宏

作用范围：当前模板。

配置步骤为：单击 Configuration→Templates，再单击模板名称（图中为 Templates OS Linuxx），最后单击 Macros，如图 5-46 所示。

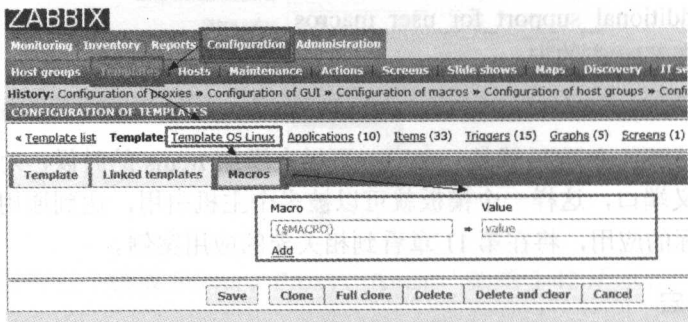


图 5-46

3. 主机宏

作用范围：当前主机。

配置步骤为：单击 Configuration→Hosts→Zabbix server→Macros，如图 5-48 所示。

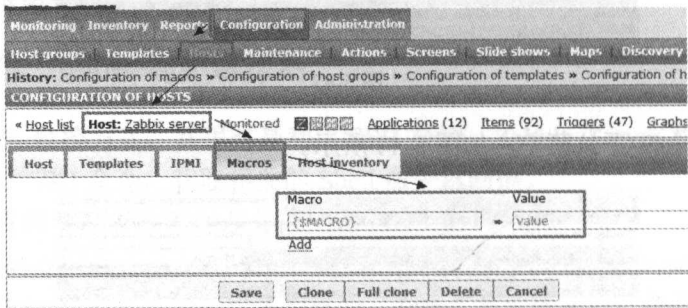


图 5-47

关于内部宏的内容，可参考以下网址。

https://www.zabbix.com/documentation/2.2/manual/appendix/macros/supported_by_location#additional_support_for_user_macros

关于主机的宏名称，其作用和注意事项如表 5-8 所示。

表 5-8

宏 名 称	作 用	注 意 事 项
{HOST.CONN}	Hostname 或者是主机的 IP 地址	{HOST.CONN<1-9>}，选择连接的方式，在 Web 界面有两个可选：IP 或 DNS
{HOST.DNS}	主机的 Hostname	{HOST.DNS<1-9>}，FQDN
{HOST.HOST}	主机的名称，Zabbix 定义的	{HOST.HOST<1-9>}，必须唯一
{HOST.IP}	主机的 IP 地址	{HOST.IP<1-9>}
{HOST.NAME}	主机可见的名称	名字显示在主机列表、map、screen 中

在 Web 中的配置内容如图 5-48 所示。

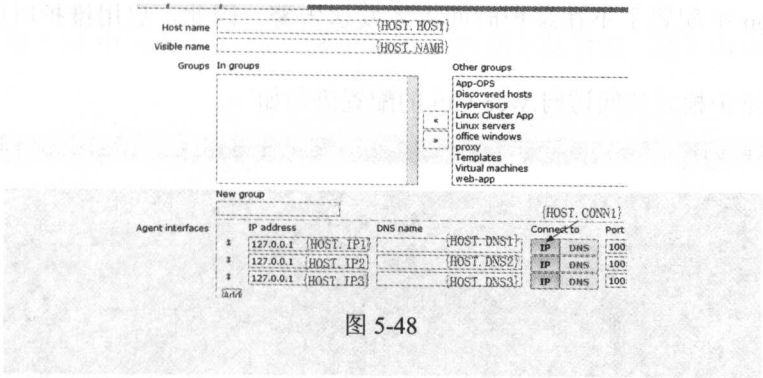


图 5-48

5.5 维护时间

在某些场合中，我们不需要进行告警，例如，业务的正常维护，所以此时维护时间功能特别有用。

配置维护时间如图 5-49 所示。

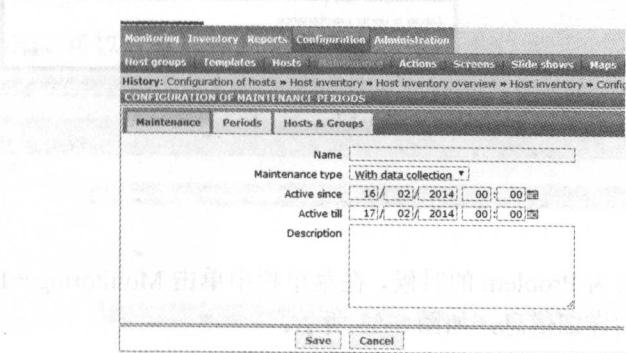


图 5-49

这里的维护时间和 Action 中的配置紧密相关，如图 5-50 所示。

图 5-50

Action 中配置了不在维护时间内才发送告警，因此，启用维护时间不会产生告警。

禁止维护模式期间访问 Web Gui 的配置语句如下。

```
shell# vim /usr/share/zabbix/conf/maintenance.inc.php #见图5-51
```

```
// Maintenance mode
define('ZBX_DENY_GUI_ACCESS', 1);
define('ZBX_DENY_GUI_ACCESS', 1);

// IP range, who are allowed to connect to FrontEnd
$ZBX_GUI_ACCESS_IP_RANGE = array('127.0.0.1');

// MSG shown on warning screen!
$_REQUEST['warning_msg'] = 'Zabbix is under maintenance.';
$_REQUEST['warning_msg'] = 'Zabbix is under maintenance.';
```

图 5-51

修改好后保存，访问显示的界面如图 5-52 所示。

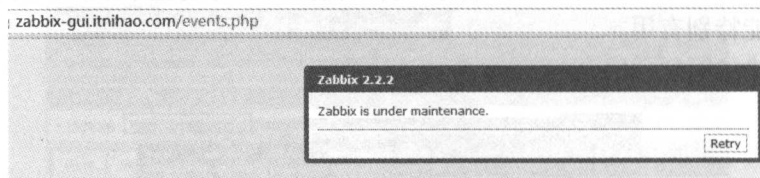


图 5-52

5.6 事件确认

当事件出现状态为 Problem 的时候，在菜单栏中单击 Monitoring→Events，在其中可以查看事件的详细信息，如图 5-53 所示。

ACK 链接有 Yes 和 No 两种状态，如图 5-54 所示。

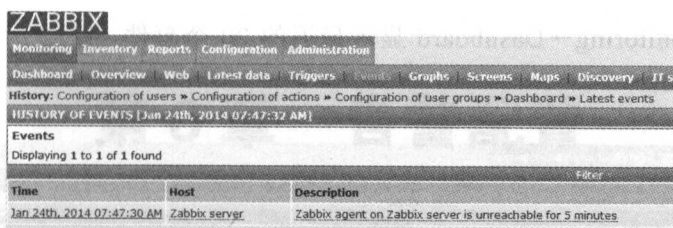


图 5-53

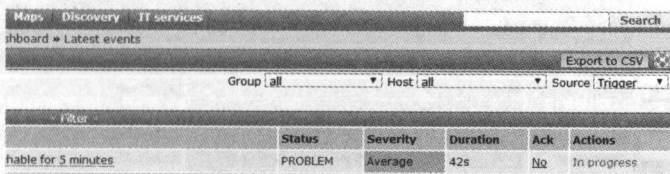


图 5-54

单击图 5-54 中 Ack 列的 No，弹出如图 5-55 所示的界面，输入内容，确认并返回结果。

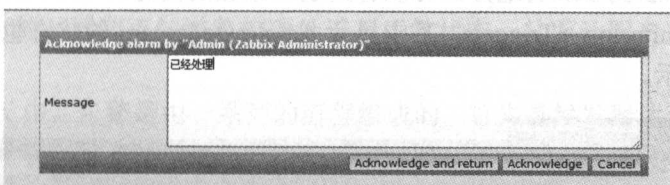


图 5-55

看到 Ack 已经变为 Yes 状态后，说明有用户修改过该状态，如图 5-56 所示。

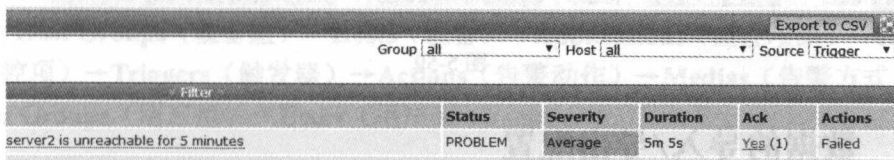


图 5-56

单击 Yes，可以查看具体的文字内容，如图 5-57 所示。

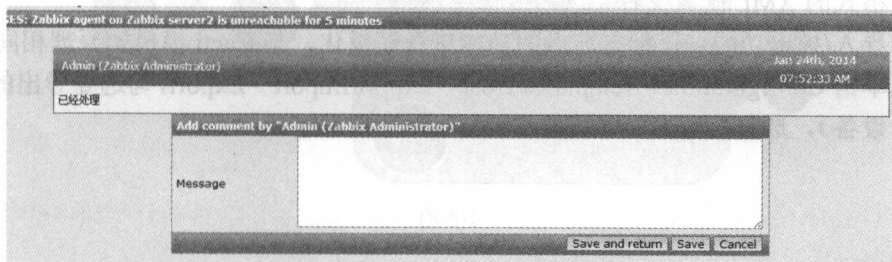


图 5-57

单击 Monitoring→Dashboard 显示最近的 20 个事件，会显示 Ack 状态，如图 5-58 所示。



图 5-58

鼠标光标放到 Yes 的地方，可以看到刚才添加的内容。

单击 Monitoring→Triggers，其中显示 Acknowledgment 的状态也会变为绿色，如图 5-59 所示。

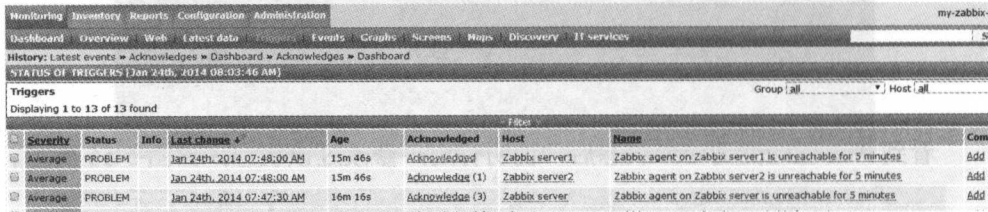


图 5-59

5.7 数据的导入/导出配置

Zabbix 提供将所有的配置导出为标准 XML 格式的文件，同样，也支持导入标准格式的 XML 配置文件。

导入/导出功能可以把之前做的功能进行模板化，与 Cacti 模板的原理相同。

单击 Configuration→Templates/Host→Export/Import→Export，勾选要导出的模板（设备），选择 Export，即可导出 XML 文件到本地。

第 6 章 告警配置

在 Zabbix 的学习过程中，告警配置成了学习的难点。本章就是为了解决这个难点的。下面将详细介绍 Trigger 的配置、Action 的配置，重点介绍 Trigger 的规则表达式、告警、升级告警、自定义告警脚本，配有大量的实例，以帮助读者深入理解这部分内容，从而解决读者对告警配置使用的困扰。

6.1 告警概述

告警是监控的重要职能，是指将达到某一阈值事件的消息发送给用户，让用户在事件发生的时候即可知道监控项处于不正常状态，从而让用户来决定是否采取相关措施。

在 Zabbix 中，告警是由一系列的流程组成的，首先是触发器达到阈值，接下来是 Action 对事件信息进行处理，其中包括两部分：第一部分是发送消息，即将告警信息发送给用户；第二部分是执行命令，即将事件用命令进行处理，达到对事件故障自动尝试恢复的效果。

Zabbix 的告警流程如图 6-1 所示，具体可表示为：

Host Groups（设备组）→ Hosts（设备）→ Applications（监控项组）→ Items（监控项）→ Triggers（触发器）→ Actions（告警动作）→ Medias（告警方式）→ User Groups（用户组）→ Users（用户）

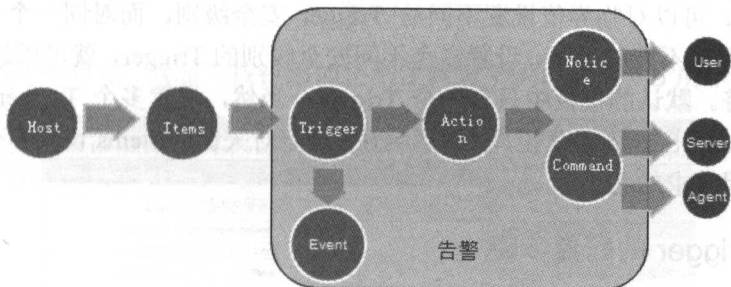


图 6-1

Zabbix 告警的配置步骤如下。

- ① 设置 Trigger。

- ② 配置用户。
- ③ 配置告警介质。
- ④ 设置 Action。

6.2 Trigger 的配置

通过前面的学习，我们知道 Items 的作用是采集数据，而不是判断采集到的数据是否属于正常值；Trigger 的作用是对采集到的数据进行阈值状态的判断，触发阈值，则会产生一个事件，同时，Action 对达到条件的 Trigger 触发告警动作。

6.2.1 Trigger 的状态

Trigger 的状态如表 6-1 所示。

表 6-1

值	描 述
OK	正常状态（老版本中是 FALSE）
PROBLEM	有事件发生，例如，CPU 负载过高（老版本中是 TRUE）

Trigger 的状态在 Zabbix-Server 每次接收到 Items 的新数据时，就会对 Items 的值进行判断（和 Trigger 的正则表达式进行条件比较）。

对于 Trigger 中的时间函数 `nodata()`、`date()`、`dayofmonth()`、`dayofweek()`、`time()`、`now()`，Zabbix-Server 会每隔 30s 进行重新判断。

每一个 Trigger 必须对应一个 Items，但一个 Items 可以对应多个 Trigger，对 Items 设置 Trigger 是非必需的，因为对某些采集数据，我们并不需要产生告警，所以就不必配置 Trigger。

Trigger 可以对临界值设置不同的 Trigger 安全级别。而对同一个 Items，在 Trigger 临界值不同的时候，设置多个不同安全级别的 Trigger，就可以达到分故障级别的告警。默认的模板中只有一个 Trigger，当然，设置多个 Trigger 时，势必会增加 Zabbix-Server 的负担，从而影响性能，但对关键的 Items 设置多个 Trigger，有实际使用需求的必要。

6.2.2 Trigger 的配置步骤

Trigger 的配置是通过单击菜单栏中的 Configuration → Hosts/template → Triggers → Create trigger 来完成的。

这里以配置一个用户登录的触发器为例进行介绍，如图 6-2 所示。

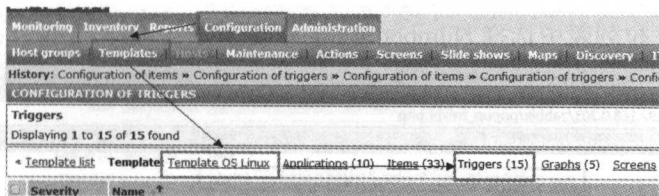


图 6-2

单击如图 6-3 所示的 Create trigger，弹出如图 6-4 所示的窗口。

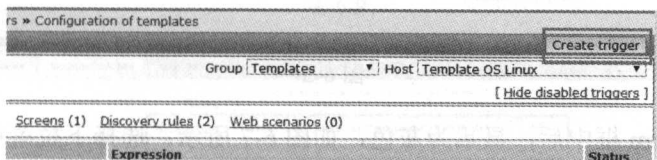


图 6-3

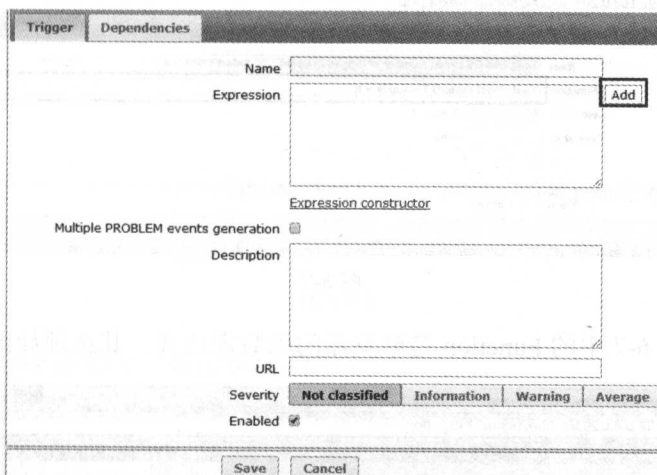


图 6-4

在图 6-4 中单击“Add”按钮添加正则逻辑表示式，如图 6-5 所示。

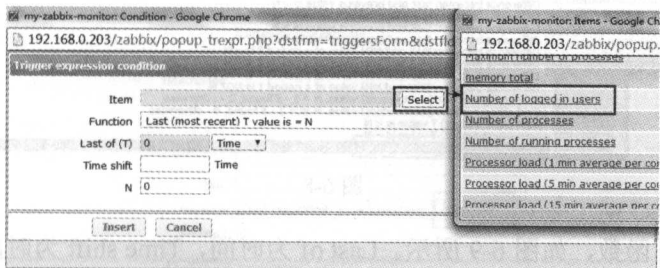


图 6-5

在 Item 下拉列表中选择 Number of logged in users，如图 6-6 所示。

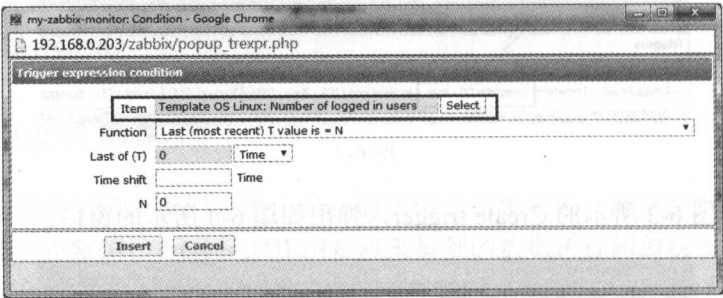


图 6-6

填充到 Item 框中后，显示为灰色，如图 6-7 所示，选择下拉菜单。

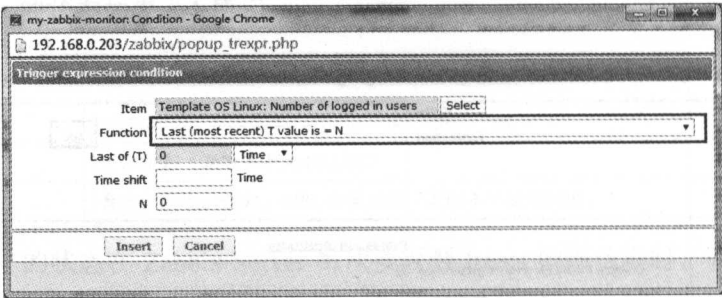


图 6-7

注意，图 6-7 中的 Function 是触发器的函数表达式，其选项如图 6-8 所示。

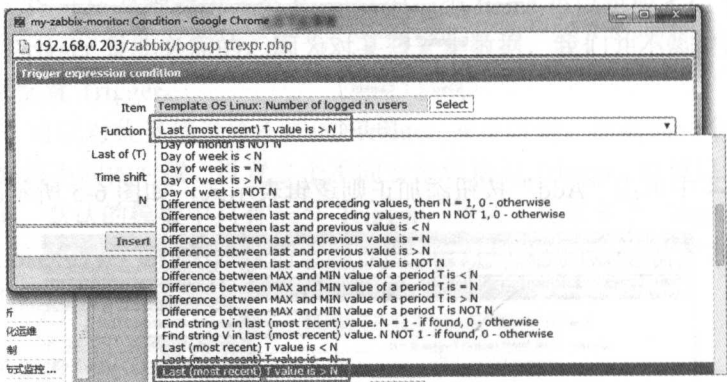


图 6-8

选择 Last 函数，如图 6-9 所示。Last of 为时间，Time shift 为时间偏移，N 为触发器的条件判断值，配置后的选项如图 6-10 所示。

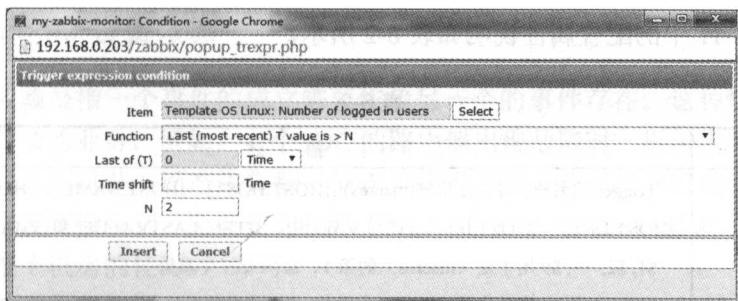


图 6-9

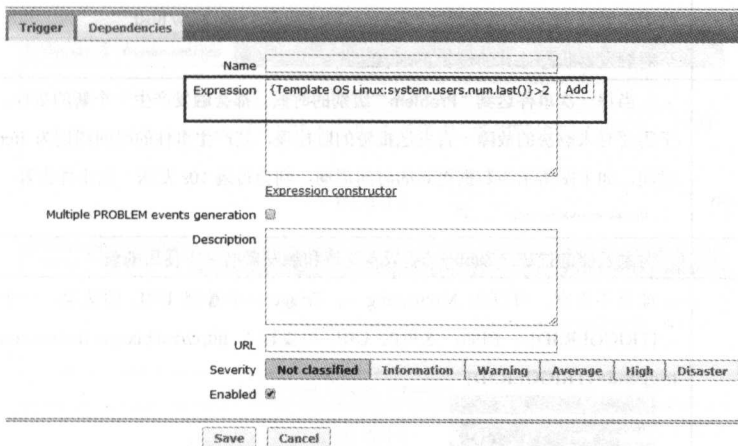


图 6-10

添加 Trigger 的名称，如图 6-11 所示。

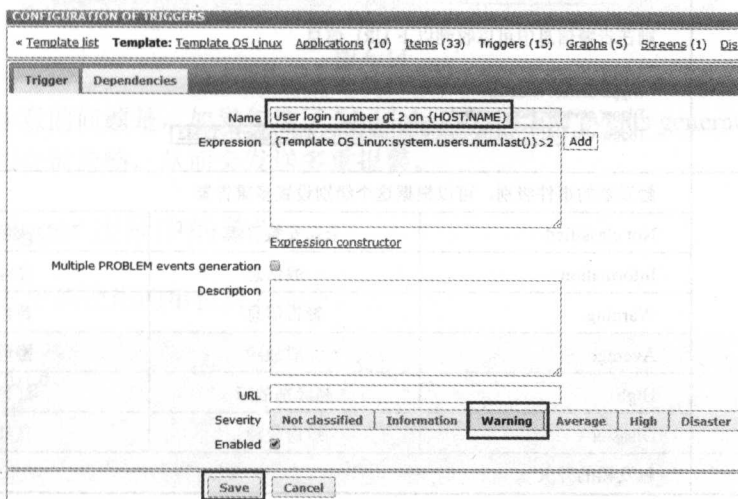
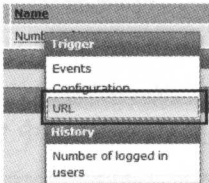


图 6-11

对图 6-11 中的配置属性说明如表 6-2 所示。

表 6-2

参 数	描 述		
Name	<p>Trigger 的名称, 可以支持宏(macros): {HOST.HOST}、{HOST.NAME}、{HOST.CONN}、{HOST.DNS}、{HOST.IP}、{ITEM.VALUE}、{ITEM.LASTVALUE}和{\$MACRO}</p> <p>\$1, \$2, ..., \$9 用于宏 (macros) 的第 1、2、...、9 个参数</p> <p>注意: \$1~\$9 参数会被解析为对应的值, 例如, 触发器的名称是“Processor load above \$1 on {HOST.NAME}”, 如果正则表达式为 {New host:system.cpu.load[percpu,avg1].last(0)}>5, 将会在触发 Trigger 后变成“Processor load above 5 on New host”</p>		
Expression	对触发器状态计算的逻辑正则表达式		
Multiple PROBLEM events generation	<p>当每一次事件达到“Problem”级别的时候, 都会触发产生一个新的事件, 该功能可用于需要对未解决的故障一直发送报警的监控项, 其产生事件的时间周期为 Items 中的更新时间, 如下图所示为数据更新的时间周期, 则会每隔 10s 发送一次事件告警</p> <p>Update interval (in sec) <input type="text" value="10"/></p>		
Description	对触发器的描述, Zabbix 2.2 版本支持和触发器名字中使用的宏		
URL	<p>如果不为空, 可以在 Monitoring → Triggers 中看到 URL 的选项。一个可用的宏为 - {TRIGGER.ID}., 例如, 这里的 URL 可以输入 http://zabbix-gui.itnihao.com/events.php?triggerid={TRIGGER.ID}</p> <div></div> <p>则在告警信息中可以收到以下 URL 信息</p> <p>Trigger: Number of logged in users gt 2 Trigger status: PROBLEM Trigger severity: Not classified Trigger URL: http://zabbix-gui.itnihao.com/events.php?triggerid=13983</p>		
Severity	触发器的事件级别, 可以根据这个级别设置多重告警		
	Not classified	未知安装等级	灰色
	Information	一般信息	亮绿
	Warning	警告信息	黄色
	Average	一般故障	橙色
	High	高级别故障	红色
	Disaster	致命故障	亮红
Enabled	触发器的开关		

6.2.3 Trigger 告警依赖

告警依赖是指一个事件的成立需要依赖另一个的事件存在。这种情况适合于逻辑比较复杂的业务，例如，一个 IDC 的路由器出现故障时，机房内所有的机器都会因为状态不可获取而产生告警，但作为管理人员，并不想同时接收所有的故障告警，只需要接收到一条有效的告警“XXX 的 IDC 机房路由器 X 发生故障，将会影响整个机房的使用。”即可。

下面的示例（如图 6-12、图 6-13 所示）是登录用户大于 2，且磁盘 I/O 过高时，可达到告警的触发条件。

图 6-12

图 6-13

配置后的界面如图 6-14 所示。

图 6-14

需要注意的问题是，如果勾选了 Multiple PROBLEM events generation 选项，则条件依赖会被忽略，从而会发送多重报警。

6.2.4 Trigger 正则中的单位

Trigger 中所使用的单位符号如下。

S: seconds
m: minutes
h: hours
d: days
w: weeks

计量单位如下。


```
K: kilo
M: mega
G: giga
T: tera
P: peta
E: exa
Z: zetta
Y: yotta
```

例如，以下语句

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>120
{host:system.uptime[] .last(0)}<86400
{host:system.cpu.load.avg(600)}<10
```

可以写成:

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>2m
{host:system.uptime.last(0)}<1d
{host:system.cpu.load.avg(10m)}<10
```

6.2.5 Trigger 表达式举例

在触发器中使用表达式是对 Items 的取值进行逻辑判断。

触发器的使用规则如下:

```
{<server>:<key>.<function>(<parameter>)}<operator><constant>
```

举例如下:

```
{Zabbix server:agent.ping.nodata(5m)}=1
{Zabbix server:agent.version.diff(0)}>0
```

1. function

触发器函数的内容参考收集到的数据，用当前时间和其他因素共同完成数据值判断。Web 界面的函数选择框如图 6-15 所示，在 6.2.6 节中，我们将对这些函数进行详细介绍。

```
Day of week is < N
Day of week is <= N
Day of week is = N
Day of week is > N
Day of week is NOT N
Difference between last and preceding values, then N = 1, 0 - otherwise
Difference between last and preceding values, then N NOT 1, 0 - otherwise
Difference between last and previous value is < N
Difference between last and previous value is = N
Difference between last and previous value is > N
Difference between last and previous value is NOT N
Difference between MAX and MIN value of a period T is < N
Difference between MAX and MIN value of a period T is = N
Difference between MAX and MIN value of a period T is > N
Difference between MAX and MIN value of a period T is NOT N
Event ID of last log entry matching regular expression T, then N = 1, 0 - otherwise
Event ID of last log entry matching regular expression T, then N NOT 1, 0 - otherwise
Find string V in last (most recent) value. N = 1 - if found, 0 - otherwise
Find string V in last (most recent) value. N NOT 1 - if found, 0 - otherwise
Last (most recent) T value is < N
Last (most recent) T value is = N
```

图 6-15

2. function parameter

函数的形参 (argument) 指的是函数可以接收不同的参数。

在形参中, “#” 在不同的函数中具有不同的含义, 例如:

- `sum(600)` 表示最近 600 秒内获取到的数值求和。
- `sum(#5)` 表示最近获取到 5 个值的和。
- `last(#5)` 表示返回给定的第 5 个值, 时间最早的值为第一个。例如, 给定值 3、7、2、6、5, `last(#2)` 的值为 7, `last(#5)` 的值为 5。

必须给函数指定一个参数, 例如, `last (0)`, 即最后一次的值, 0 为参数。

`avg`、`count`、`last`、`min` 和 `max` 支持在某个时间段之前的数据获取, 例如, `avg(1h,1d)`, 表示 1 小时之前 1 天的平均值。

触发器表达式中使用支持单位符号, 例如, '5m' (分钟) 代替 '300s' (秒), '15' (天) 代表 '86400s' (秒)。

3. 运算符

触发器支持的运算符 (优先级渐降) 如表 6-4 所示。

表 6-3

优先级	运算符	含 义
1	/	整除 (division)
2	*	乘 (Multiplication)
3	-	减 (Arithmetical minus)
4	+	加 (Arithmetical plus)
5	<	小于 (Less than) 运算符定义为: $A < B \Leftrightarrow (A \leq B - 0.000001)$
6	>	大于 (More than) 运算符定义为: $A > B \Leftrightarrow (A \geq B + 0.000001)$
7	#	不等于 (Not equal) 运算符定义为: $A \# B \Leftrightarrow (A \leq B - 0.000001) \vee (A \geq B + 0.000001)$
8	=	等于 (Is equal) 运算符定义为: $A = B \Leftrightarrow (A > B - 0.000001) \& (A < B + 0.000001)$
9	&	逻辑与 (Logical AND)
10		逻辑或 (Logical OR)

4. 触发器示例

示例 1: 对主机 `www.zabbix.com` 负载值进行判断, 语句如下。

```
{www.zabbix.com:system.cpu.load[all,avg1].last(0)}>5
```

'{www.zabbix.com:system.cpu.load[all,avg1]}'给出了监控参数的名称。它指定服务器是'www.zabbix.com', Items 是'system.cpu.load[all,avg1]', 使用函数'last()'取最近一次获取到的值, '>5'表示来自 www.zabbix.com 主机的最后负载值大于 5 时触发器进入 PROBLEM 状态。

示例 2: www.zabbix.com 负载判断, 语句如下。

```
{www.zabbix.com:system.cpu.load[all,avg1].last(0)}>5|{www.zabbix.com:system.cpu.load[all,avg1].min(10m)}>2
```

负载大于 5 或者是最近 10 分钟的负载大于 2, 就会触发 Trigger 进入 PROBLEM 状态。

示例 3: 对文件/etc/passwd 监控。

使用函数 diff:

```
{www.zabbix.com:vfs.file.cksum[/etc/passwd].diff(0)}>0
```

当文件/etc/passwd 之前的 checksum 值与最近的值不同时, 则会触发 Trigger 进入 PROBLEM 状态。

示例 4: 对网卡流量进行条件判断。

使用函数 min:

```
{www.zabbix.com:net.if.in[eth0,bytes].min(5m)}>100K
```

当最近 5 分钟内 eth0 接收的字节数大于 100KB, 则会触发 Trigger 进入 PROBLEM 状态。

示例 5: 两个 SMTP 服务器的集群节点都停止了。

注意, 在一个表达式中使用两个不同的主机, 语句如下。

```
{smtp1.zabbix.com:net.tcp.service[smtp].last(0)}=0&{smtp2.zabbix.com:net.tcp.service[smtp].last(0)}=0
```

当 SMTP 服务器 smtp1.zabbix.com 与 smtp2.zabbix.com 都停止时, 表达式为真, 则会触发 Trigger 进入 PROBLEM 状态。

这个示例也就是告警关联, 对于避免误报具有很好的效果。

示例 6: Zabbix 客户端代理需要更新。

使用函数 str():

```
{zabbix.zabbix.com:agent.version.str("beta8")}=1
```

当 Zabbix 客户端代理有版本 beta8 时, 该表达式为真。

示例 7: 服务器不可达, 语句如下。

```
{zabbix.com:icmpping.count(30m,0)}>5
```

主机 zabbix.zabbix.com 在最近 30 分钟内超过 5 次不可达, 该表达式为真。

示例 8: 最近 3 分钟内没有回应。

使用函数 `nodata()`:

```
{zabbix.com:tick.nodata(3m)}=1
```

'tick'必须使用类型'Zabbix trapper'。为了这个触发器能工作, tick 必须定义。该主机应该使用 Zabbix-Sender 定期为该参数发送数据。如果 180s 都没有收到数据, 该触发器的值变为 **PROBLEM**。

示例 9: CPU 在夜间的活跃度。

使用函数 `time()`:

```
{zabbix:system.cpu.load[all,avg1].min(5m)}>2&{zabbix:system.cpu.load[all,avg1].time(0)}>000000&{zabbix:system.cpu.load[all,avg1].time(0)}<060000
```

触发器只在晚上(00:00—06:00)为可用, 当 5 分钟内的负载大于 2 时, 该触发器的值变为 **PROBLEM**。

示例 10: 检查客户端本地时间是否与 Zabbix-Server 服务器时间同步。

使用函数 `fuzzytime()`:

```
{MySQL_DB:system.localtime.fuzzytime(10)}=0
```

当 MySQL_DB 的本地时间与 Zabbix-Server 的时间相差超过 10s 时, 触发器变为 **PROBLEM** 状态。

5. 滞留状态

有时候, 触发器必须在不同的情况下有不同的条件。例如, 我们想定义当服务器房间的温度超过 20℃时触发器变为 **PROBLEM** 状态, 然后触发器一直停留在这个状态, 除非温度低于 15℃。为了实现这种功能, 我们定义下面的触发器。

示例 1: 服务器房间温度过高。

```
{{TRIGGER.VALUE}=0&{server:temp.last(0)}>20} |  
{{TRIGGER.VALUE}=1&{server:temp.last(0)}>15}
```

注意, 使用了一个宏{TRIGGER.VALUE}, 这个宏返回当前触发器的值。

{TRIGGER.VALUE}=0 是 OK 状态, 且值大于 20, 则触发告警。

{TRIGGER.VALUE}=1 是 **PROBLEM** 状态, 当前状态已经为 **PROBLEM**, 且值大于 15, 一直停留在这个状态, 直到值小于 15。

PROBLEM: 温度高于 20℃。

Recovery: 温度低于 15℃。

示例 2: 磁盘空间不足。

```
{{TRIGGER.VALUE}=0&{server:vfs.fs.size[/,free].max(5m)}<10G} |  
{{TRIGGER.VALUE}=1&{server:vfs.fs.size[/,free].min(10m)}<40G}
```

其中:

PROBLEM: 最近 5 分钟磁盘剩余空间少于 10GB。

Recovery: 最近 10 分钟磁盘剩余空间大于 40GB。

6.2.6 Trigger 函数

本节内容是姚炫伟（网名为@绿小小肥）翻译的，在此表示感谢。本书对其进行直接引用，目的是方便读者更好地理解，在此基础上加了一些例子作为补充说明，原文地址为：

<http://pengyao.org/zabbix-triggers-functions.html>

官方文档地址为：

<https://www.zabbix.com/documentation/2.2/manual/appendix/triggers/functions>

(1) abschange

参数：直接忽略后边的参数。

支持值类型：float、int、str、text、log。

描述：返回最近获取到的值与之前值的差值的绝对值。对于字符串类型，0 表示值相等，1 表示值不同。

例如：{server:vfs.fs.size[/,free].abschange(10m)}<10，表示在 Server 设备中，Key 值 vfs.fs.size[/,free]最近一次获取到的值和在前 10 分钟的差值为 10，其结果可能是最近一次的值比之前大，也有可能比之前的值小，即从-10 到 0，再到 10 之间的一个范围，可以叫作抖动值或者误差范围。

(2) avg

参数：秒或#num。

支持值类型：float 和 int。

描述：返回指定时间间隔的平均值。时间间隔可以通过第一个参数秒数设置或收集值的数目（需要在前边加上#，比如，#5 表示最近 5 次的值）。如果有第二个，则表示时间漂移（time shift），例如，查询一天之前一小时的平均值，对应的函数是 avg（3600,86400），时间漂移参数是 Zabbix 1.8.2 新增加的。

例如：{server:vfs.fs.size[/,free].avg(#5,10m)}<50G，表示在 Server 设备中，Key 值 vfs.fs.size[/,free]最近 10 分钟内，最近的 5 次取值的平均值小于 50GB。请注意，10 分钟内如果取值更新间隔时间为 1 分钟，则#5 代表的是第 6、7、8、9、10 分钟的数值，如果取值时间更新间隔为 2 分钟，则#5 代表的是第 2、4、6、8、10 分钟的数值。

(3) change

参数：直接忽略掉后边的参数。

支持值类型：float、int、str、text、log。

描述：返回最近获取到的值与之前值的差值（注意，与 `abschange` 函数不同，不是绝对值）。对于字符串类型，0 表示值相等，1 表示值不同。

例如：`{server:vfs.fs.size[/,free].abschange(5m)}<0`，代表最近一次的值比前 5 分钟获取的值要小。

(4) count

参数：秒或#num。

支持值类型：float、int、str、text、log。

描述：返回指定时间间隔内的数值统计。时间间隔可以通过第一个参数设置为时间或收集值的数目（需要在值前边加上#）。本函数可以支持第二个参数作为样本（pattern）数据，第三个参数作为操作（operator）参数，第四个参数作为时间漂移（time shift）参数。对于样本，监控项的值为整数（integer）时使用精确匹配，监控项的值为浮点型（float）时允许偏差 0.0000001。

支持的操作（operators）类型如下。

- eq：相等。
- ne：不相等。
- gt：大于。
- ge：大于或等于。
- lt：小于。
- le：小于或等于。
- like：内容匹配。

对于整数和浮点型监控项目，支持 eq（默认）、ne、gt、ge、lt、le；对于 string、text、log 监控项，支持 like（默认）、eq、ne。

示例：

- `count(600)`：最近 10 分钟值的个数。
- `count(600,12)`：最近 10 分钟值等于 12 的个数。
- `count(600,12,"gt")`：最近 10 分钟值大于 12 的个数。
- `count(#10,12,"gt")`：最近的 10 个值中，值大于 12 的个数。
- `count(600,12,"gt",86400)`：24 小时之前的前 10 分钟数据中，值大于 12 的个数。
- `count(600,,,86400)`：24 小时之前的前 10 分钟数据值的个数。

#num 参数从 Zabbix 1.6.1 开始支持，time shift 参数和字符串操作从 Zabbix 1.8.2 开始支持。

(5) date

参数：直接忽略后边的参数。

支持值类型：所有 (any)。

描述：返回当前日期（格式为 YYYYMMDD），例如，20140701。

(6) dayofmonth

参数：直接忽略后边的参数。

支持值类型：所有 (any)。

描述：返回当前是本月第几天（数值范围为 1~31），该函数从 Zabbix 1.8.5 开始支持。

(7) dayofweek

参数：直接忽略后边的参数。

支持值类型：所有 (any)。

描述：返回当前是本周的第几天（数值返回为 1~7），星期一是 1，星期天是 7。

(8) delta

参数：秒或#num。

支持值类型：float 和 int。

描述：返回指定时间间隔内的最大值与最小值的差值 (max()-min())。时间间隔作为第一个参数，可以是秒或者收集值的数目。从 Zabbix 1.8.2 开始，支持可选的第二个参数 time_shift。

例如：{Switch:net.if.in[eth0].delta(10m)}>10M，表示设备 Switch 的接口 eth0 在最近 10 分钟内，最大值和最小值之间的差大于 10MB，即认为故障发生。

(9) diff

参数：忽略。

支持值类型：float、int、str、text 和 log。

描述：返回值为 1，表示最近的值与之前的值不同，0 为其他情况。

例如：{Windows:agent.version.diff(0)}>0，表示设备 Windows 的 agent.version 在最近一次的值和之前的值不同。

(10) fuzzytime

参数：秒。

支持值类型：float 和 int。

描述：返回值为 1，表示监控项值的时间戳比 Zabbix-Server 的时间多 N 秒，0 为其他情况。常使用 system.localtime 来检查本地时间是否与 Zabbix-server 的时间相同。

(11) iregexp

参数：第一个为字符串，第二个为秒或#num。

支持值类型: str、log、text。

描述: 与 `regexp` 类似, 区别是不区分大小写。

(12) last

参数: 秒或#num。

支持值类型: float、int、str、text 和 log。

描述: 最近的值, 如果为秒, 则忽略, #num 表示最近第 *N* 个值。注意, 当前的#num 和其他一些函数的#num 的意思是不同的。

示例:

- `last(0)` 等价于 `last(#1)`。
- `last(#3)` 表示最近获取的监控项值的第 3 个值 (并不是最近的三个值), 如获取到的值分别是 A、B、C 三个值, 从时间顺序来看, A 为第一个值, B 为第二值, #3 就代表第 3 个值 C。

本函数也支持第二个参数 `time_shift`, 例如: `last(0,86400)` 返回一天前的最近值。如果在 `history` 中, 同一秒内有多个值存在, Zabbix 不保证值的精确顺序。

#num 从 Zabbix 1.6.2 开始支持, `timeshift` 从 Zabbix 1.8.2 开始支持, 可以查询 `avg()` 函数获取它的使用方法。

例如: `{MySQL:mysql.ping.last(#3,5m)}=0`, 表示设备 MySQL 在最近 5 分内的第 3 次取值结果为 0, 即认为故障发生。注意, #3 的用法和其他函数不同。

(13) logeventid

参数: string。

支持值类型: log。

描述: 检查最近的日志条目的 Event ID 是否匹配正则表达式。参数为正则表达式、POSIX 扩展样式。当返回值为 0 时, 表示不匹配, 1 表示匹配。该函数从 Zabbix 1.8.5 开始支持。

(14) logseverity

参数: 忽略。

支持值类型: log。

描述: 返回最近日志条目的日志等级 (log severity)。当返回值为 0 时, 表示默认等级, *N* 为具体对应的等级 (整数, 常用于 Windows event logs)。Zabbix 日志等级来源于 Windows event log 的 Information 列。

(15) logsource

参数: string。

支持值类型: log。

描述: 检查最近的日志条目是否匹配参数的日志来源。当返回值为 0 时, 表示不匹配, 1 表示匹配。通常用于 Windows event logs 监控, 例如, `logsource["VMWare`

Server"]。

(16) max

参数：秒或#num。

支持值类型：float 和 int。

描述：返回指定时间间隔的最大值。时间间隔作为第一个参数，可以是秒或收集值的数目（前缀为#）。从 Zabbix 1.8.2 开始，函数支持第二个可选参数 time_shift，可以查看 avg() 函数获取它的使用方法。

例如：{ftpserver:net.tcp.service[ftp].max(#3)}=0，表示设备 ftpserver 中的 key 为 net.tcp.service[ftp] 在最近 3 次获取到的最大数值都为 0，则视为故障。

(17) min

参数：秒或#num。

支持值类型：float 和 int。

描述：返回指定时间间隔的最小值。时间间隔作为第一个参数，可以是秒或收集值的数目（前缀为#）。从 Zabbix 1.8.2 开始，函数支持第二个可选参数 time_shift，可以查看 avg() 函数获取它的使用方法。

例如：{gateway:icmppingloss.min(5m)}>20，表示 gateway 设备在 5 分钟内用 icmppingloss 获取到的最小值大于 20，即故障发生。

(18) nodata

参数：秒。

支持值类型：any。

描述：当返回值为 1 时，表示指定的间隔（间隔不应小于 30 秒）没有接收到数据，0 表示其他情况。

例如：{v.itnihao.com:agent.ping.nodata(5m)}=1，表示设备 v.itnihao.com 的 agent.ping 在最近 5 分钟内没有接收到数据，即故障发生。

(19) now

参数：忽略。

支持值类型：any。

描述：返回距离 Epoch（1970 年 1 月 1 日 00:00:00 UTC）时间的秒数。

(20) prev

参数：忽略。

支持值类型：float、int、str、text 和 log。

描述：返回之前的值，类似于 last(#2)。

(21) regexp

参数：第一个参数为 string，第二个参数为秒或#num。

支持值类型：str、log、text。

描述: 检查最近的值是否匹配正则表达式, 参数的正则表达式为 POSIX 扩展样式, 第二个参数为秒数或收集值的数目, 将会处理多个值。本函数区分大小写。当返回值为 1 时, 表示找到, 0 为其他情况。

(22) str

参数: 第一个参数为 string, 第二个参数为秒或#num。

支持值类型: str、log、text。

描述: 查找最近值中的字符串。第一个参数指定查找的字符串, 大小写敏感。第二个可选的参数指定秒数或收集值的数目, 将会处理多个值。当返回值为 1 时, 表示找到, 0 为其他情况。

例如: {Tomcat:jmx["Catalina:type=ProtocolHandler,port=8080",compression].str(off)} = 1, 表示设备 Tomcat 从 key 中获取到了字符串 off。

(23) strlen

参数: 秒或#num。

支持值类型: str、log、text。

描述: 指定最近值的字符串长度 (并非字节), 参数值类似于 last 函数。例如, strlen(0)等价于 strlen(#1), strlen(#3)表示最近的第三个值, strlen(0,86400)表示一天前最近的值。该函数从 Zabbix 1.8.4 开始支持。

(24) sum

参数: 秒或#num。

支持值类型: float 和 int。

描述: 返回指定时间间隔中收集到的值的总和。时间间隔作为第一个参数, 支持秒或收集值的数目 (以#开始)。从 Zabbix 1.8.2 开始, 本函数支持 time_shift 作为第二个参数。可以查看 avg 函数获取它的用法。

(25) time

参数: 忽略。

支持值类型: any。

描述: 返回当前时间, 格式为 HHMMSS, 例如, 123055。

6.3 添加 Actions

6.3.1 Actions 概述

如果想在产生事件后, 即当触发器条件被满足时, 采取一些操作, 比如, 发送事件通知、远程执行命令等, 则需要配置 Actions。Actions 可以对如图 6-16 所示的类型产生的事件进行响应, 其中, 各参数的含义如表 6-4 所示。



图 6-16

表 6-4

名 称	作 用
Trigger events	当 Trigger 的状态从 OK 改变为 PROBLEM
Discovery events	当 network discovery 工作
Auto registration events	主动模式的 Agent 自动注册
Internal events	当 Items 变成不被支持(unsupported)或者 Trigger 变成未知状态(unknown state)，该功能需 Zabbix 2.2 以上版本支持

6.3.2 Actions 的配置

1. 配置步骤

- ① 在菜单栏中单击 Configuration→Actions。
- ② 在 Event source 下拉菜单中选择事件来源。
- ③ 单击 Create action。
- ④ 设置 Action 参数。
- ⑤ 单击 Conditions 按钮，设置 Action 的依赖条件。
- ⑥ 单击 Operation 按钮，设置执行动作。

通过菜单 Configuration→Actions→Create Action 来创建 Actions, 如图 6-17 所示。
表 6-5 对 Actions 界面的各项参数进行了详细说明。

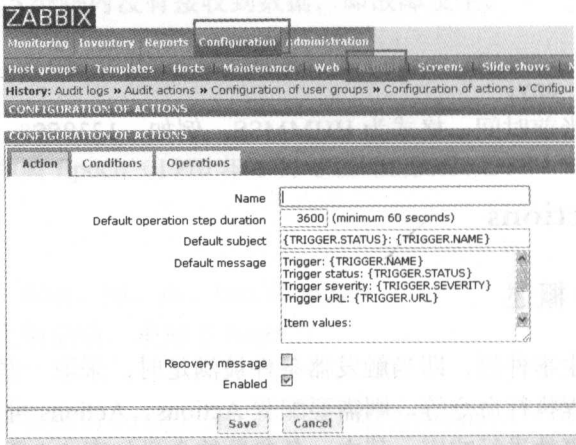


图 6-17

表 6-5

参 数	描 述
Name	唯一的 action 名称
Default subject	默认消息主题，可以包含宏（macros），如：{TRIGGER.STATUS}: {TRIGGER.NAME}
Default message	默认消息，可以包含宏（macros），如：{TRIGGER.NAME}、{TRIGGER.STATUS}、{TRIGGER.SEVERITY}、{TRIGGER.URL}、{ITEM.NAME1}、{HOST.NAME1}、{ITEM.KEY1}、{ITEM.VALUE1}、{EVENT.ID}、{EVENT.DATE}、{EVENT.TIME}、{TIME}等
Recovery message	事件恢复的消息，即事件从状态 Problem 变成 OK 状态，仅会发送一次消息，可自定义主题和内容。可用宏为：{EVENT.ACK.HISTORY}、{EVENT.ACK.STATUS}、{EVENT.*}、{EVENT.RECOVERY.*}、{EVENT.RECOVERY.DATE}、{EVENT.RECOVERY.TIME}等
Recovery subject	恢复消息的主题，可以包含宏（macros）
Enabled	勾选开启这个 Action，不勾选则关闭

2. 发送消息的宏配置

宏的配置是在消息主题和消息内容中完成的。

(1) 示例 1

消息：

```
{TRIGGER.NAME}: {TRIGGER.STATUS}
```

当接收到消息后，内容将会变为：

```
Processor load is too high on server zabbix.zabbix.com: PROBLEM
```

(2) 示例 2

消息：

```
Processor load is: {zabbix.zabbix.com:system.cpu.load[,avg1].last(0)}
```

当接收到消息后，内容将会变为：

```
Processor load is: 1.45
```

(3) 示例 3

消息：

```
Latest value: {{HOST.HOST}}:{{ITEM.KEY}}.last(0)}  
MAX for 15 minutes: {{HOST.HOST}}:{{ITEM.KEY}}.max(900)}  
MIN for 15 minutes: {{HOST.HOST}}:{{ITEM.KEY}}.min(900)}
```

当接收到消息后，内容将会变为：

```
Latest value: 1.45  
MAX for 15 minutes: 2.33  
MIN for 15 minutes: 1.01
```


(4) 示例 4

支持在消息内容中显示触发器的正则表达式。

消息:

```
Trigger: {TRIGGER.NAME}
Trigger expression: {TRIGGER.EXPRESSION}

1. Item value on {HOST.NAME1}: {ITEM.VALUE1} ({ITEM.NAME1})
2. Item value on {HOST.NAME2}: {ITEM.VALUE2} ({ITEM.NAME2})
```

当接收到消息后, 内容将会变为:

```
Trigger: Processor load is too high on a local host
Trigger expression: {Myhost:system.cpu.load[percpu,avg1].last(0)}
>5 | {Myotherhost:system.cpu.load[percpu,avg1].last(0)}>5

1. Item value on Myhost: 0.83 (Processor load (1 min average per core))
2. Item value on Myotherhost: 5.125 (Processor load (1 min average per core))
```

(5) 示例 5

消息:

```
Problem:
Event ID: {EVENT.ID}
Event value: {EVENT.VALUE}
Event status: {EVENT.STATUS}
Event time: {EVENT.TIME}
Event date: {EVENT.DATE}
Event age: {EVENT.AGE}
Event acknowledgement: {EVENT.ACK.STATUS}
Event acknowledgement history: {EVENT.ACK.HISTORY}

Recovery:
Event ID: {EVENT.RECOVERY.ID}
Event value: {EVENT.RECOVERY.VALUE}
Event status: {EVENT.RECOVERY.STATUS}
Event time: {EVENT.RECOVERY.TIME}
Event date: {EVENT.RECOVERY.DATE}
```

当接收到消息后, 内容将会变为:

```
Problem:
Event ID: 21874
Event value: 1
Event status: PROBLEM
Event time: 13:04:30
Event date: 2014.01.02
Event age: 5m
Event acknowledgement: Yes
Event acknowledgement history: 2014.01.02 13:05:51 "John Smith (Admin)"

Recovery:
```

```

Event ID: 21896
Event value: 0
Event status: OK
Event time: 13:10:07
Event date: 2014.01.02

```

6.3.3 Conditions 的配置

1. 条件之间的逻辑运算符

运算符类型如图 6-18 所示，其中，各选项的含义如下。

- **AND**: 所有的条件必须同时满足。
- **OR**: 满足条件中的一个即可。
- **AND/OR**: 两个关系的组合，AND用于不同条件的关系，OR用于相同条件的关系。

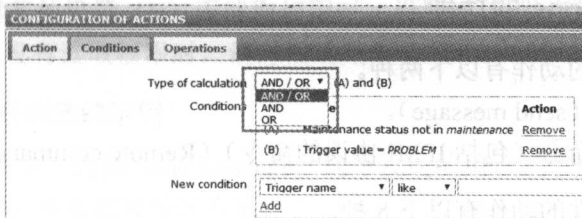


图 6-18

AND/OR 的示例如下：

```

Host group = Oracle servers
Host group = MySQL servers
Trigger name like 'Database is down'
Trigger name like 'Database is unavailable'

```

换成表达式为：

```

(Host group = Oracle servers or Host group = MySQL servers) and (T
rigger name like 'Database is down' or Trigger name like 'Database is
unavailable')

```

Host group 和 Trigger name 同时存在的条件下才能成立。

条件匹配运算符说明如表 6-6 所示。

表 6-6

运 算 符	描 述	运 算 符	描 述
=	条件等于	like	部分匹配
>=	条件大于或等于	not like	部分不匹配
<=	条件小于或等于	in	在范围内
<>	排除	not in	不在范围内

2. 触发器的状态

如果 Trigger 的状态从 OK 变成 PROBLEM, Trigger 的状态将会变成 PROBLEM。

如果 Trigger 的状态从 PROBLEM 变成 OK, Trigger 的状态将会变成 OK。

如果 Trigger 的状态既不为 OK, 也不为 PROBLEM, 则其状态为未知状态, 即 UNKNOWN。

当对 Triggers 创建一个新的 Action 时, 会自动添加以下两个条件 (用户可以删除或添加更多的条件)。

① “Trigger value = PROBLEM”: 任何问题都会发送消息, 这意味着配置一个 Action 时, 如果没有设置条件, 将会发送所有的事件消息。

② “Maintenance status = not in maintenance”: 在维护时间段将不会发送消息。

6.3.4 Operations 的功能

触发器事件的动作有以下两种。

- 发送消息 (send message)。
- 执行远程命令 (包括 IPMI 协议的命令) (Remote command)。

自动发现事件的动作有以下 8 种。

- 增加主机 (add host)。
- 删除主机 (remove host)。
- 开启主机监控 (enable host)。
- 关闭主机监控 (disable host)。
- 增加到组 (add to group)。
- 从一个组中删除 (delete from group)。
- 链接到模板 (link to template)。
- 取消模板链接 (unlink from template)。

自动注册事件的动作有以下 6 种。

- 发送消息 (send message)。
- 远程命令 (remote command)。
- 增加主机 (add host)。
- 关闭主机监控 (disable)。
- 增加到组 (add to group)。
- 链接到模板 (link to template)。

6.3.5 告警消息发送的配置

当出现 Trigger 状态改变时, 应发送消息通知相关人员。

1. 配置发送消息的步骤

配置发送消息的步骤如下。

- ① 配置消息发送介质 (Media)，步骤为：单击 Administration→Media types。
- ② 配置一个对需要发送消息设备的用户，步骤为：单击 Administration→Users→Create User。
- ③ 配置 Actions 中的消息发送。

2. 发送消息的权限问题

Zabbix 发送告警的消息时，需要发送消息的账户对主机所产生的事件具有读的权限，对 Trigger 的表达式有访问的权限。例如，默认是 admin 用户，可以对所有的机器有读取的权限，而对于自己添加的用户 (user)，如果只对部分机器 (A) 有读权限，而对其他机器 (B) 无读取权限时，设置对 B 发送消息，而选择的用户 (user) 对 B 没有读取权限时，消息发送则不会成功。

3. 配置发送消息的示例

如图 6-19 所示，选择 Send message，选择发送的用户或者用户组，两者可以同时选择，另外，还要选择发送介质。

Default operation step duration: 3600 (minimum 60 seconds)

Steps	Details	Start in	Duration (sec)	Action
No operations defined.				

Operation details

Step: From 1 To 1 (0 - infinitely) Step duration 0 (minimum 60 seconds, 0 - use action default)

Operation type: Send message

Send to User groups: User group Zabbix administrators Add

Send to Users: User Admin (Zabbix Administrator) Add

Send only to: All

Default message: ☒

Conditions: No conditions defined.

Operation condition: Event acknowledged [v] [v] [Not Ack] Add Cancel

图 6-19

4. 发送消息的日志

在发送告警之后，很多时候需要查看日志，如图 6-20 所示，具体步骤如下。

- ① 单击 Monitoring→Events，查看事件。
- ② 单击 Administration→Audit，查看 Action 消息。

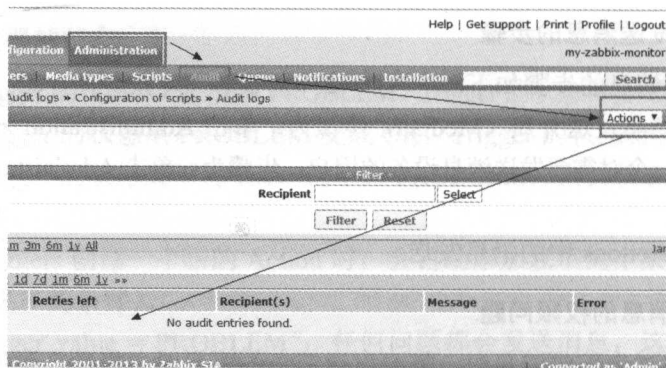


图 6-20

6.3.6 执行远程命令的配置

通过远程命令可以做到以下操作：

- 若服务无响应，自动重启应用（Web 服务、中间件、CRM）。
- 通过 IPMI 协议远程重启服务器。
- 当磁盘满后，清空磁盘无用的文件，以释放空间。
- 当 CPU 负载过高，从一个物理机迁移 VM 到另一台物理机。
- 当 CPU 资源不足时，为云环境增加一个节点资源 CPU（磁盘、内存及其他）等。

远程命令和发送消息类似，只是操作的是执行一个命令，而不是发送消息。

要注意以下几点：

- 远程命令不支持主动模式的 Agent，是从 Server 向 Agent 执行。
- 远程命令不支持代理模式。
- 命令的长度被限制为 255 个字符。
- 可以在一个操作中执行多条命令，添加新的远程命令即可。
- 远程命令可以包含宏。
- Zabbix 用户必须对该命令具有执行的权限，如果没有权限，需配置 sudo，让 Zabbix 程序在执行命令的过程中，使用 sudo 无密码的方式，以 ROOT 身份，执行具有 ROOT 权限的命令。
- Agent 的防火墙允许从数据包进来的连接。
- Zabbix 不会检测命令是否执行成功，只是命令执行。

1. 远程命令的配置

前提条件是 Agent 开启支持远程命令的参数。

```
EnableRemoteCommands=1
```

默认是不支持远程命令的执行的。修改此参数后，需要重启 Agent 进程服务。配置步骤如下（见图 6-21）。

图 6-21

- ① 单击 Configuration→Actions。
- ② 在 Operations 选项中，选择远程命令（Remote command）的操作类型（operation type）。
- ③ 选择远程命令类型（IPMI、Custom script、SSH、Telnet、Global script）。
- ④ 输入执行的命令。

2. 通过远程执行命令对 Apache 进行重启

如果 Agent 端的 Apache 无响应，在 Agent 端对 Apache 进行重启操作即可。

```
sudo /etc/init.d/apache restart
```

步骤如下。

- ① Agent 端配置执行远程命令。

```
shell# vim /etc/zabbix/zabbix_agentd.conf
EnableRemoteCommands=1
```

- ② 配置 sudo。

```
shell# visudo
# 允许 'zabbix' 用户重启 Apache 不需要密码
zabbix ALL=NOPASSWD: /etc/init.d/apache restart
```


③ 配置条件。

配置条件包括：应用分组是 Apache；不在维护期；触发器事件为 PROBLEM；触发器等级为严重时执行，如图 6-22 所示。

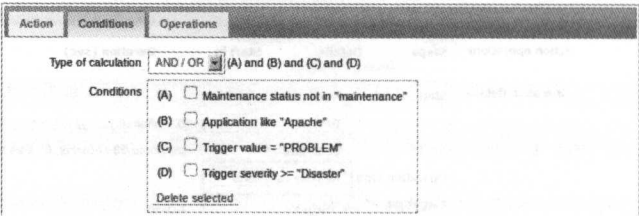


图 6-22

④ 配置远程命令。

如图 6-23 所示，配置远程命令的执行。

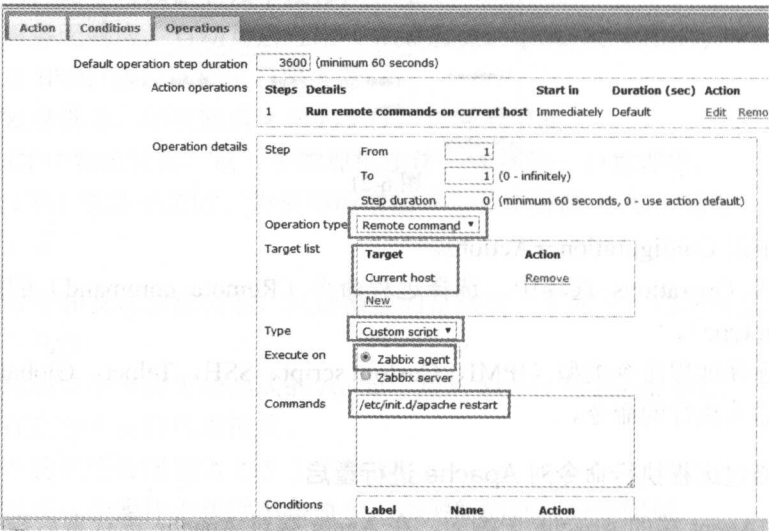


图 6-23

6.4 邮件告警配置的实例

6.4.1 创建 Media

步骤为：登录 Zabbix Web 页面，单击 Administration→Media types→Create Media Type（右上角）。

Create Media Type 参数设置的说明如表 6-7 所示。

表 6-7

参 数	描 述	参 数	描 述
Name	Media 类型的名称	SMTP helo	设置正确的 SMTP helo 值，通常是一个域名
Type	选择类型为 Email	SMTP email	设置发送消息的 Email 账号
SMTP server	设置 SMTP 服务器来处理传出消息		

如图 6-24 所示，此处采用本机的 postfix (sendmail) 发送邮件，postfix 用系统默认安装的即可，无须其他设置。采用这种方式的邮件发送是无法使用认证的，如需要认证，则可以采用脚本方式。

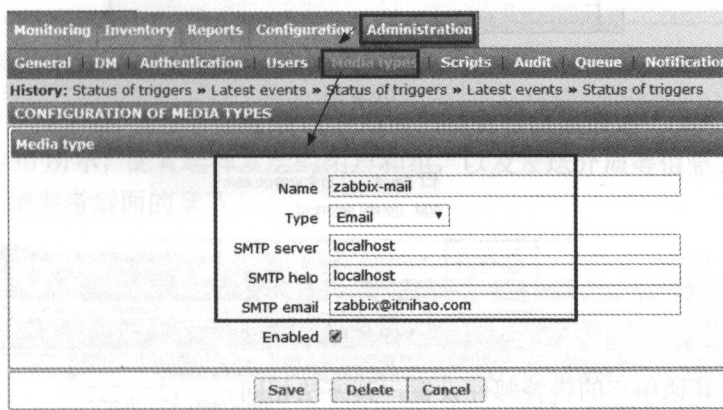


图 6-24

6.4.2 创建用户

创建用户的步骤为：单击 Administration→Users→Create User，如图 6-25 所示。

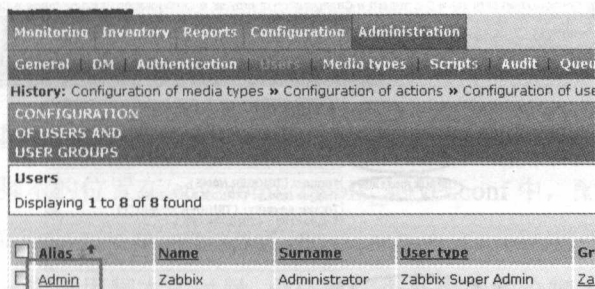


图 6-25

用户可以配置 Media，发送告警的方式，如邮件、短信等，如图 6-26 所示。

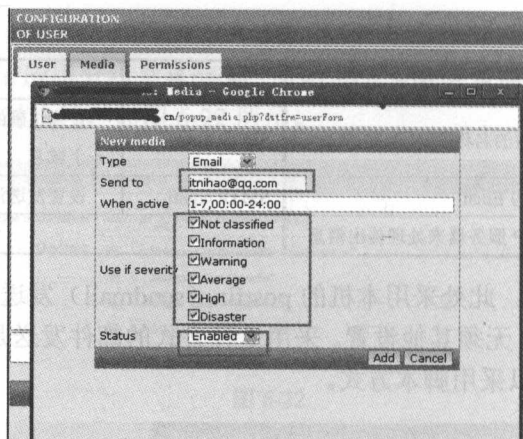


图 6-26

单击“Add”按钮添加后保存，如图 6-27 所示。

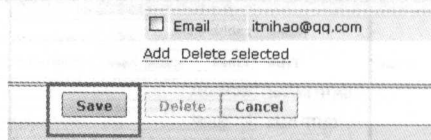


图 6-27

如果是其他用户的告警邮件设置，其方法相同。

6.4.3 创建 Actions

Actions 的创建步骤为：单击 Configuration→Actions→Create actions，如图 6-28 所示。

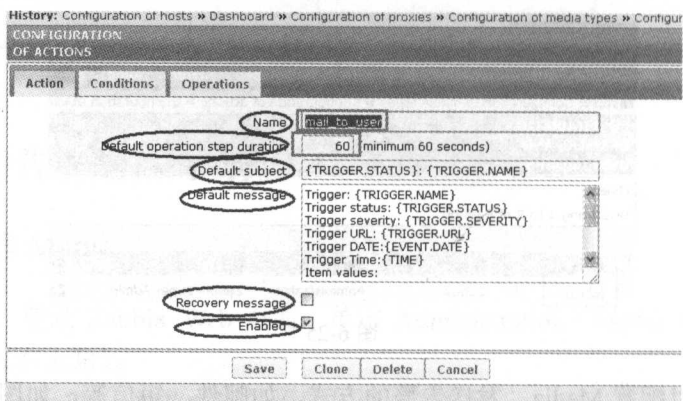


图 6-28

单击 Conditions，如图 6-29 所示为默认的配置参数。

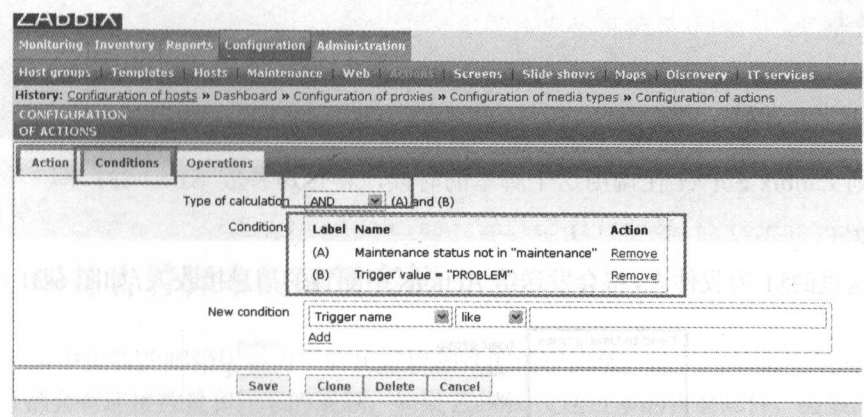


图 6-29

如图 6-30 所示，配置选择发送的用户和组，以及发送介质等信息。如需配置告警升级，请参考后面的章节。

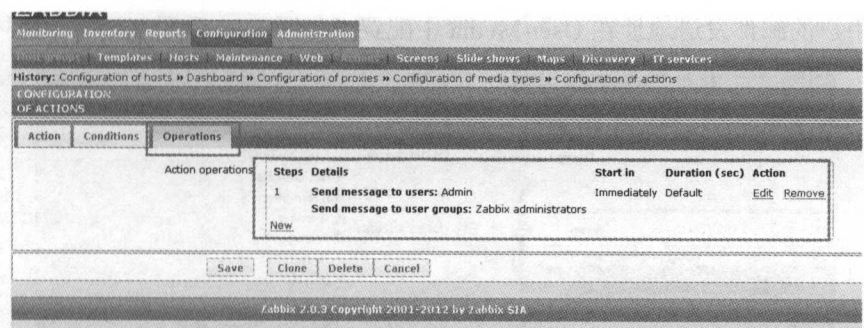


图 6-30

6.5 自定义脚本告警

1. 自定义脚本告警原理

自定义告警脚本的位置在/etc/zabbix/zabbix_server.conf中，配置语句如下。

```
AlertScriptsPath=/etc/zabbix/alertscripts
```

Zabbix-Server 在调用脚本的时候，会传递三个变量参数给脚本作为位置参数：
\$1、\$2、\$3。其中，\$1 表示收件人；\$2 表示主题；\$3 表示内容。

例如，/etc/zabbix/alertscripts/scripts.sh 脚本内容为：

```
#!/bin/bash
```

```

to=$1
subject=$2
body=$3

cat <<EOF | mail -s "$subject" "$to"
$body
EOF

```

则 Zabbix-Server 在调用这个脚本的时候，将传递参数 \$1 \$2 \$3，即

```
/etc/zabbix/alertscripts/scripts.sh $1 $2 $3
```

这里的 \$1 为收件人，将会发送给 Actions 中配置的消息接收人，如图 6-31 所示。

Send to User groups	User group	Action
	Zabbix administrators	Remove
	Add	

Send to Users	User	Action
	Add	

Send only to: - All -

图 6-31

用户的邮件发送地址在 User-Media 中配置，如图 6-32 所示。

图 6-32

所以，\$1 的值最终为 User→Media 菜单栏下 Send to 中填入的地址，\$2 为 Actions 中配置的默认主题，\$3 为 Actions 中配置的默认消息，如图 6-33 所示。

图 6-33

掌握以上原理后，自定义脚本就很容易了。例如，将消息通过短信猫、短信接口等发送。

2. 短信接口告警脚本

短信接口发送脚本语句如下。

```
#!/bin/bash
curl http://X.X.X.X/smsapi/user=$1&subject=$2&content=$3&key=XXX
```

6.6 邮件告警脚本的配置实例

网络中免费的邮箱服务为了防止垃圾邮件的情况出现，出于对邮箱安全的考虑，大部分都会有连接数量和频率的限制，如果 Zabbix 发送过多的告警邮件，会被邮件服务器当作垃圾，从而被拒绝发送。因此，请读者尽可能用自建的邮件服务器发送信息。

1. Zabbix-Server 自定义告警脚本配置

修改/etc/zabbix/zabbix_server.conf 配置，语句如下。

```
shell#vim/etc/zabbix/zabbix_server.conf
AlertScriptsPath=/etc/zabbix/alertscripts/
shell#mkdir -p /etc/zabbix/alertscripts/
shell#cat /etc/zabbix/alertscripts/zabbix_sendmail.py
#!/usr/bin/python
#coding:utf-8

import smtplib
from email.mime.text import MIMEText
import os
import argparse
import logging
import datetime

mail_host = 'smtp.163.com'
mail_user = 'monitor_itnihao'
mail_pass = 'my_password'
mail_postfix = '163.com'

def send_mail(mail_to,subject,content):
    me = mail_user+"<" + mail_user+"@" + mail_postfix+">"
    msg = MIMEText(content)
    msg['Subject'] = subject
    msg['From'] = me
    msg['to'] = mail_to
    global sendstatus
    global senderr

    try:
        smtp = smtplib.SMTP()
```



```

        smtp.connect(mail_host)
        smtp.login(mail_user,mail_pass)
        smtp.sendmail(me,mail_to,msg.as_string())
        smtp.close()
        print 'send ok'
        sendstatus = True
    except Exception,e:
        senderr=str(e)
        print senderr
        sendstatus = False

def logwrite(sendstatus,mail_to,content):
    logpath='/var/log/zabbix/alert'

    if not sendstatus:
        content = senderr

    if not os.path.isdir(logpath):
        os.makedirs(logpath)

    t=datetime.datetime.now()
    daytime=t.strftime('%Y-%m-%d')
    daylogfile=logpath+'/'+str(daytime)+'.log'
    logging.basicConfig(filename=daylogfile,level=logging.DEBUG)
    logging.info('*'*130)
    logging.debug(str(t)+' mail send to {0},content is :\n {1}'.format(mail_to,content))

    if __name__ == "__main__":
        parser = argparse.ArgumentParser(description='Send mail to user for zabbix alerting')
        parser.add_argument('mail_to',action="store", help='The address of the E-mail that send to user ')
        parser.add_argument('subject',action="store", help='The subject of the E-mail')
        parser.add_argument('content',action="store", help='The content of the E-mail')
        args = parser.parse_args()
        mail_to=args.mail_to
        subject=args.subject
        content=args.content

        send_mail(mail_to,subject,content)
        logwrite(sendstatus,mail_to,content)

```

注意：上面的这个脚本文件需要 Zabbix 用户具有执行权限，以确保脚本能正常运行。下面对脚本文件进行权限改变。

```

shell# chmod 700 /etc/zabbix/alertscripts/zabbix_sendmail.py
shell# chown zabbix.zabbix /etc/zabbix/alertscripts/zabbix_sendmail.py

```

如果是 RHEL 6.4 以上版本的系统，会缺少 python-argparse 模块，请安装 python-argparse，软件包在本书的 github 中，网址为：

<https://github.com/itnihao/zabbix-book/tree/master/06-chapter>

```
shell# rpm -ivh python-argparse-1.2.1-2.el6.noarch.rpm
shell# python /etc/zabbix/alertscripts/zabbix_sendmail.py info@itnihao.com test "test to send mail"
```

如果 info@itnihao.com 能收到邮件，说明脚本发送邮件成功，否则，说明配置存在错误，或者邮箱不支持此种邮件发送类型。

上面这个脚本的特点是将发送的所有信息都以日志方式保存，即按天存储，这样可以很方便地知道告警消息是否发送成功。

对于前面所提到的，一个机房由于网络抖动，而引起的大规模误报，其处理思路除了设置复杂依赖关系的触发器，还可以在告警脚本中进行改良，例如，对上面脚本的告警主题和告警内容进行条件判断，对重复发送的消息进行隔离或丢弃，或者将告警结果发送给第三方告警分析系统，从而起到告警汇集的作用，有效地减少误报。

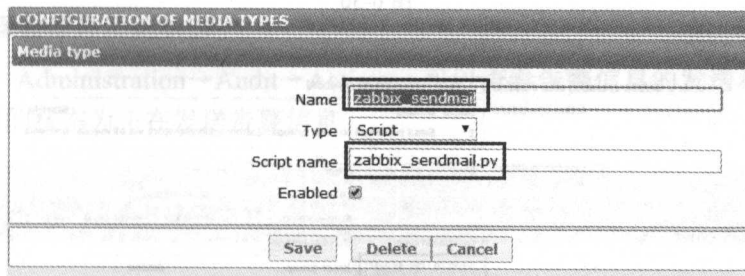
2. 重启 Zabbix-Server 服务

修改完配置文件后，需要重启 Zabbix-Server 服务，语句如下：

```
shell# service zabbix-server restart
```

3. Web GUI 配置自定义脚本的步骤

配置步骤为：单击 Configuration→Media types→Create media type，出现的界面如图 6-34 所示。



CONFIGURATION OF MEDIA TYPES

Media type

Name: zabbix_sendmail

Type: Script

Script name: zabbix_sendmail.py

Enabled: ☒

Save Delete Cancel

图 6-34

添加触发设置：单击 Configuration→Actions→Create action，如图 6-35 所示。

添加告警发送的条件，如图 6-36 所示。

选择告警的条件，读者可根据需要选择多种方式的告警条件，如图 6-37 所示。

选择发送的媒介为脚本，选择发送用户。发送告警的用户必须对告警信息所在的机器有读取权限，否则，告警信息无法正常发送。

CONFIGURATION OF ACTIONS

Action **Conditions** **Operations**

Name: Report problems to Zabbix administrators

Default subject: {TRIGGER.STATUS}: {TRIGGER.NAME}

Default message: Trigger: {TRIGGER.NAME}
Trigger status: {TRIGGER.STATUS}
Trigger severity: {TRIGGER.SEVERITY}
Trigger URL: {TRIGGER.URL}

Item values:

Recovery message: ☒

Recovery subject: {TRIGGER.STATUS}: {TRIGGER.NAME}

Recovery message: Trigger: {TRIGGER.NAME}
Trigger status: {TRIGGER.STATUS}
Trigger severity: {TRIGGER.SEVERITY}
Trigger URL: {TRIGGER.URL}

Item values:

Enabled: ☒

Save Clone Delete Cancel

图 6-35

CONFIGURATION OF ACTIONS

Action **Conditions** **Operations**

Type of calculation: AND / OR (A) and (B) and (C)

Conditions

Label	Name	Action
(A)	Maintenance status not in maintenance	Remove
(B)	Trigger value = PROBLEM	Remove
(C)	Trigger severity = Disaster	Remove

New condition: Trigger severity = Not classified

Add

Save Clone Delete Cancel

图 6-36

CONFIGURATION OF ACTIONS

Action **Conditions** **Operations**

Default operation step duration: 3600 (minimum 60 seconds)

Action operations

Steps	Details	Start in
1	Send message to user groups: Zabbix administrators via all media	Immedia

Operation details

Step: 1

From: 1

To: 1 (0 - infinitely)

Step duration: 0 (minimum 60 seconds, 0 - use action)

Operation type: Send message

Send to User groups

User group	Action
Zabbix administrators	Remove
Add	

Send to Users

User	Action
Admin (Zabbix Administrator)	Remove
Add	

Send only to: zabbix_sendmail

Default message: ☒

Conditions

Label	Name	Action
New		

Update Cancel

Save Clone Delete Cancel

图 6-37

4. 配置用户的邮件发送

步骤为：单击 Administration→Users→User name→Media，如图 6-38 所示。

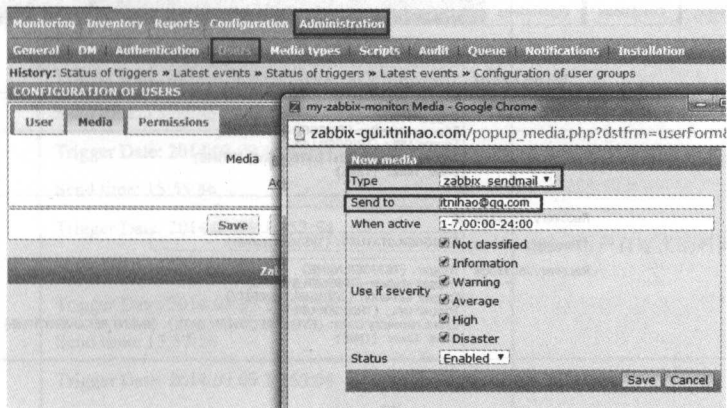


图 6-38

图 6-38 中，各项的含义如下。

- Type: 选择以哪种方式发送告警。
- Send to: 收件人。
- When active: 消息发送的时间段。
- Use if severity: 消息发送级别。
- Status: 开关。

5. 查看邮件发送状态

单击 Administration→Audit→Actions，可以查看告警信息的发送状态，如图 6-39 所示的状态为正在发送告警信息。

Time	Type	Status	Retries left	Recipient(s)	Message
09 Mar 2014 09:57:51	zabbix_sendmail	sent		itnihao@qq.com	Subject: PROBLEM: Number of logged in users gt 2

图 6-39

6.7 告警升级的机制

告警升级可以对告警结果按自定义的时间段进行消息发送，并执行命令，形成一个梯度的告警处理。

在 Action 中配置的步骤如图 6-40 所示。

在发送的消息中添加事件发生的时间戳和发送时间，便于了解事件发生的时间，语句如下。

Trigger Date: {EVENT.DATE} {EVENT.TIME}
Send Time: {TIME}

CONFIGURATION OF ACTIONS

Action **Conditions** **Operations**

Name: Escalations

Default subject: {TRIGGER.STATUS}: {TRIGGER.NAME}

Default message: Trigger: {TRIGGER.NAME}
Trigger status: {TRIGGER.STATUS}
Trigger severity: {TRIGGER.SEVERITY}
Trigger URL: {TRIGGER.URL}
Trigger Date: {EVENT.DATE} {EVENT.TIME}
Send Time: {TIME}

Recovery message: ☒

Recovery subject: {TRIGGER.STATUS}: {TRIGGER.NAME}

Recovery message: Trigger: {TRIGGER.NAME}
Trigger status: {TRIGGER.STATUS}
Trigger severity: {TRIGGER.SEVERITY}
Trigger URL: {TRIGGER.URL}
Event recovery Date: {EVENT.RECOVERY.DATE} {EVENT.RECOVERY.TIME}
Send Time: {TIME}

Enabled ☒

图 6-40

示例 1：告警升级的配置如图 6-41 所示。

Action **Conditions** **Operations**

Default operation step duration: 120 (minimum 60 seconds)

Action operations

Steps	Details	Start in	Duration (sec)	Action
1 - 2	Send message to users: Admin (Zabbix Administrator) via zabbix_sendmail	Immediately	60	Edit Remove
3 - 5	Send message to users: web1 Song (Song wu Mr Song) via zabbix_sendmail	00:02:00	60	Edit Remove
6 - 7	Send message to users: Admin (Zabbix Administrator) via zabbix_sendmail	00:05:00	Default	Edit Remove

New

图 6-41

告警信息将分为三个等级梯度发送，如图 6-42 所示。

- 第一梯度：1、2 步，发送给 Admin 用户，间隔时间为 60s，发送两次，直到故障恢复。
- 第二梯度：3、5 步，发送给 web1 Song 用户，间隔时间为 60s，发送 3 次，直到故障恢复。
- 第三梯度：6、7 步，发送给 Admin 用户，间隔时间为 120s，即默认的间隔时间，发送两次，直到故障恢复。

1 120 (minimum 60 seconds)

3

Steps	Details	Start in	Duration (sec)
1 - 2	Send message to users: Admin (Zabbix Administrator) via zabbix_sendmail	Immediately	60
3 - 5	Send message to users: web1 Song (Song wu Mr Song) via zabbix_sendmail	00:02:00	60
6 - 7	Send message to users: Admin (Zabbix Administrator) via zabbix_sendmail	00:05:00	Default

New

图 6-42

用户收到的告警信息时间如表 6-8 所示。

表 6-8

用户 (steps)	事件接收时间为	时间 间隔
Admin (1、2)	Trigger Date: 2014.03.09 15:53:54 Send time: 15:53:56	60s, 用户自定义的时间间隔
	Trigger Date: 2014.03.09 15:53:54 Send time: 15:54:56	
web1 Song (3~5)	Trigger Date: 2014.03.09 15:53:54 Send time: 15:55:56	60s, 用户自定义的时间间隔
	Trigger Date: 2014.03.09 15:53:54 Send time: 15:56:56	
	Trigger Date: 2014.03.09 15:53:54 Send time: 15:57:56	
Admin (6、7)	Trigger Date: 2014.03.09 15:53:54 Send Time: 15:58:56	120s, 使用默认的时间间隔
	Trigger Date: 2014.03.09 15:53:54 Send Time: 16:00:56	

示例 2：告警升级的配置如图 6-43 所示。

120 (minimum 60 seconds)			
Steps	Details	Start in	Duration (sec)
1 - 0	Send message to users: Admin (Zabbix Administrator) via zabbix_sendmail	Immediately	60
5 - 7	Send message to users: web1 Song (Song wu Mr Song) via zabbix_sendmail	00:04:00	90
New			

图 6-43

在图 6-43 中，用户 Admin 每隔 60s 发送一次告警信息，直到事件状态变成 OK 为止。用户 web1 Song 在故障 4 分钟后，每隔 90s 发送一次告警信息，共计发送 3 次。

示例 3：告警升级的配置如图 6-44 所示。

Action	Conditions	Operations		
Default operation step duration <div>1800</div> (minimum 60 seconds)				
Action operations				
Steps	Details	Start in	Duration (sec)	Action
1 - 0	Send message to user groups: MySQL Administrators via Email	Immediately	Default	Edit Remove
5	Send message to user groups: Database manager via Email	02:00:00	Default	Edit Remove
6	Run remote commands on current host	02:30:00	Default	Edit Remove
7	Send message to user groups: Guests via Email	03:00:00	Default	Edit Remove
9	Run remote commands on current host	04:00:00	Default	Edit Remove
New				

图 6-44

- 用户 MySQL Administrations 每隔半个小时发送一次告警。
- 用户 Database manager 在事件发生后的 2 小时发送告警信息。
- 在 2 小时 30 分钟后，执行远程重启 MySQL 的命令。
- 在 3 小时后，即执行远程命令后并未解决问题，发送告警消息给用户 Guests。
- 在 4 小时后，通过 IPMI 对机器执行远程重启命令。

示例 4：告警升级的配置如图 6-45 所示。

Action	Conditions	Operations		
Default operation step duration 1800 (minimum 60 seconds)				
Action operations				
Steps	Details	Start in	Duration (sec)	Action
1 - 4	Send message to user groups: MySQL Administrators via Email	Immediately	Default	Edit Remove
5 - 6	Send message to user groups: Database manager via Email	02:00:00	3600	Edit Remove
5 - 7	Send message to user groups: Zabbix administrators via Email	02:00:00	600	Edit Remove
11	Send message to user groups: Guests via Email	04:00:00	Default	Edit Remove
New				

图 6-45

假如事件故障在 00:00 时刻发生，那么

- 用户 MySQL administrators 在事件发生后的 0:00、0:30、1:00、1:30 时间内，将会收到告警信息。
- 用户 Database manager 在 2:00 和 2:10 时间内收到告警信息。（这里为什么不是 3:00？看到 5、6 步骤，设置被后面的 5、7 步骤的 600s 替代。）
- 用户 Zabbix administrators 在 2:00、2:10、2:20 时间内收到告警信息，时间间隔为 600s。
- 用户 guest 在 4:00 受到告警信息。（步骤 8~11 是用默认的时间间隔半个小时。）

综上所述，当 Steps 设置为从 1 到 0 时，会一直发送告警信息，直到事件状态变成 OK，当 Steps 设置为从 1 到 1 时，则只会发送一次告警，后面不会继续发送告警信息。

6.8 告警配置故障排查

告警达到触发器设置的条件时，在事件中也可以看到，但未收到告警信息。影响告警发送的条件如下。

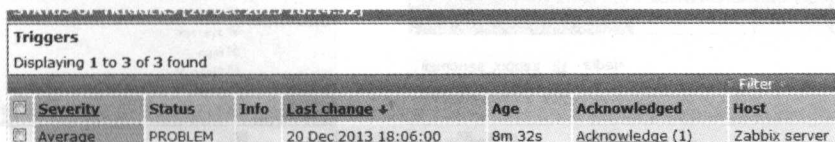
- 告警脚本权限、告警账户是否配置正确。
- 用户对发送告警信息的机器是否有可读权限，这一点非常重要。
- 告警信息的接收方是否拒绝接收告警信息，在某些情况下，告警信息会被当作垃圾信息给屏蔽了。

- 告警配置是否正确，是否达到了触发器设置的阈值，Action 中的条件是否正确，是否在维护状态。

如何查看告警发送失败的日志呢？单击 Administration→Audit→Actions，在这里可以看到告警信息的日志。

1. 告警信息未发送的示例

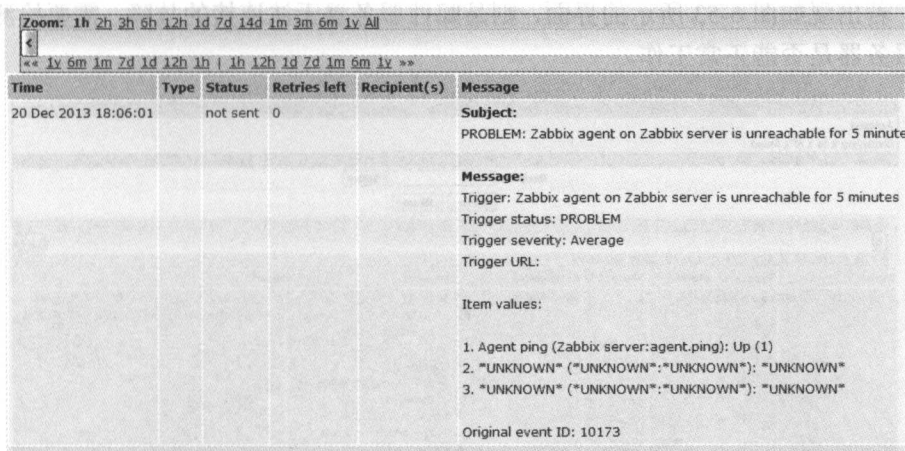
如图 6-46 所示，在 Monitoring→Triggers 中看到故障已经发生，但是为何没有发送告警呢？



Severity	Status	Info	Last change +	Age	Acknowledged	Host
Average	PROBLEM		20 Dec 2013 18:06:00	8m 32s	Acknowledge (1)	Zabbix server

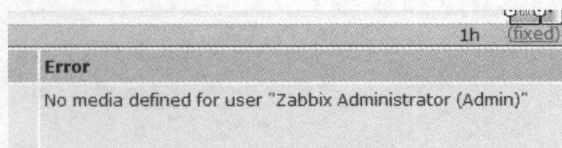
图 6-46

查看日志（单击 Administration→Audit→Actions），状态未发送，如图 6-47 所示。原因是用户没有定义发送介质，如图 6-48 所示。



Time	Type	Status	Retries left	Recipient(s)	Message
20 Dec 2013 18:06:01		not sent	0		Subject: PROBLEM: Zabbix agent on Zabbix server is unreachable for 5 minute Message: Trigger: Zabbix agent on Zabbix server is unreachable for 5 minutes Trigger status: PROBLEM Trigger severity: Average Trigger URL: Item values: 1. Agent ping (Zabbix server:agent.ping): Up (1) 2. "UNKNOWN" ("UNKNOWN": "UNKNOWN"): "UNKNOWN" 3. "UNKNOWN" ("UNKNOWN": "UNKNOWN"): "UNKNOWN" Original event ID: 10173

图 6-47



Error
No media defined for user "Zabbix Administrator (Admin)"

图 6-48

若要解决这个问题，给用户定义发送介质即可，如图 6-49 至图 6-52 所示。

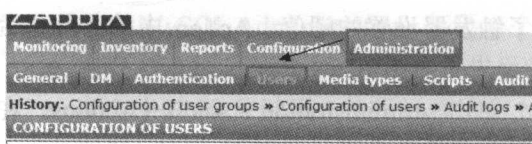


图 6-49

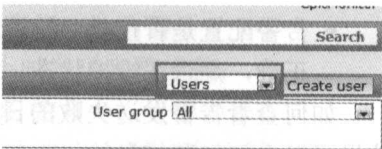


图 6-50

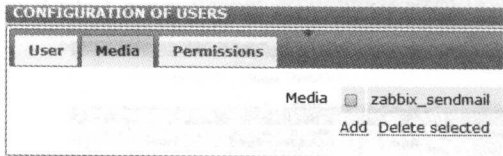


图 6-51

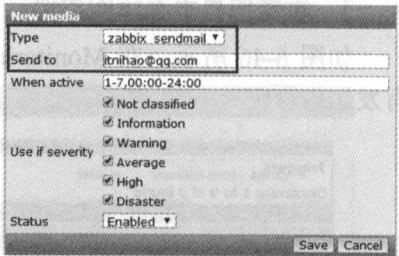


图 6-52

解决这个问题后，告警即可发送成功。

2. 邮件服务器连接失败示例

若出现如图 6-53 所示的界面，则是邮件服务器无法连接的故障，需要检测邮件服务器是否能正常工作。

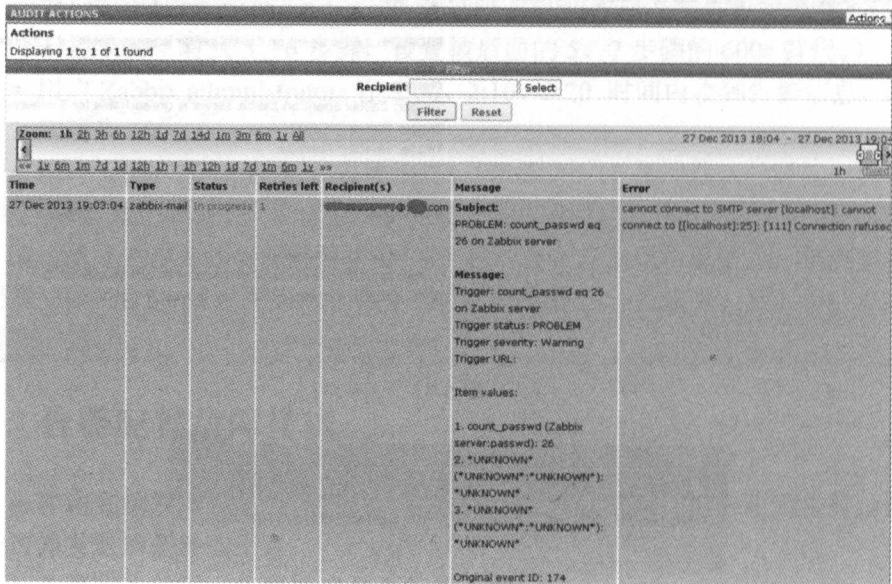


图 6-53

第2部分

中级部分

第 7 章 监控方式剖析

Zabbix 自带多种类型的监控方式，大致分两类：公共的协议和 Zabbix 专业的协议。支持多种协议监控的方式，相应地，也支持多种设备的监控，从而可以对复杂的网络环境进行监控。

7.1 Zabbix 支持的监控方式

在 Zabbix 的 Web 界面中，可以对 Agent、SNMP、JMX、IPMI 这四种协议的可用状态进行显示。处于正确状态的监控会显示绿色图标（如图 7-1 所示），红色代表当前不可用。如出现故障，将鼠标指针移动到相应的状态，就会提示具体的故障信息，如图 7-2 所示。

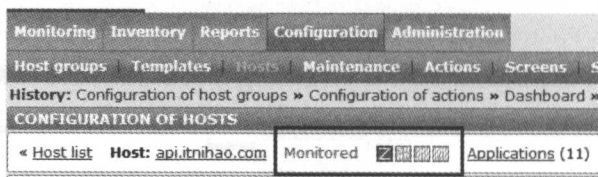


图 7-1

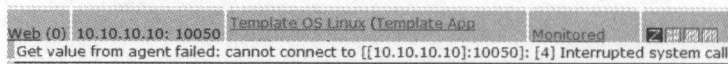


图 7-2

当然，Zabbix 可以支持的监控方式还有很多，如图 7-3 所示。

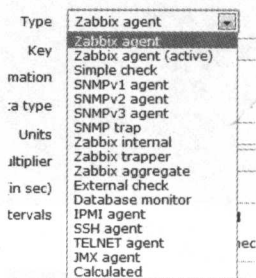


图 7-3

在第5章已介绍了大部分监控方式，本章将对部分监控方式进行深入探讨。

7.2 Zabbix 监控方式的逻辑

图 7-4 展示了 Agent、SNMP、JMX 和 IPMI，这四种监控方式的逻辑。

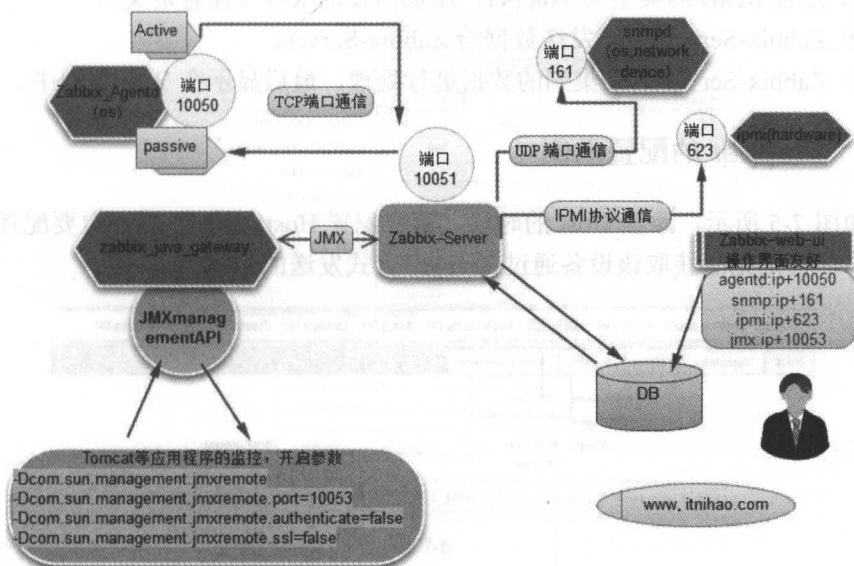


图 7-4

对于其他的监控方式，其原理与图 7-4 类似。

7.3 Agent 监控方式

Agent 分为主动和被动两种方式，关于 Agent 的监控，第 5 章进行了详细讲解，在第 8 章中，将对主动和被动两种工作模式进行详细讲解。

7.4 Trapper 监控方式

Trapper 是发送任意的数据给 Zabbix-Server，采用主动推送的方式，与主动模式有一些相同，但又有区别，其区别在于 Trapper 不需要安装客户端，Key 的名称是可以灵活定义的，这意味着 Zabbix-Server 可以获取需要的任意数据。

在 Trapper 的工作模式中，Zabbix 数据发送的程序是 zabbix-sender。

7.4.1 Trapper 的配置步骤

Trapper 的配置步骤如下。

- ① 添加主机，需要配置 Host name（Trapper 的数据处理方式为主机名，而不是 IP 地址）。
- ② 配置 Items 的类型为 Trapper，添加需要的 Key（任意定义）。
- ③ Zabbix-Sender 程序发送数据给 Zabbix-Server。
- ④ Zabbix-Server 对采集到的数据进行处理，最后显示在 Web 界面中。

7.4.2 Trapper 的配置示例

如图 7-5 所示，添加 Host 的时候，需要配置 Host name，且一定要配置唯一的名称，否则无法获取该设备通过 Trapper 方式发送的数据。

The screenshot shows the Zabbix web interface for configuring a host. The 'Host' tab is selected, and the host name is 'trapper'. The 'Agent interfaces' section is expanded, showing the 'IP address' field with the value '192.168.0.200'. The 'Status' is set to 'Monitored'. The 'Save' button is at the bottom.

图 7-5

注意，Agent interfaces 中的 IP address 可以随便填一个，但是为了识别是哪台机器，方便管理人员查看，需填写真实的 IP 地址，Zabbix-Server 并不会主动去连接这个 IP 地址，而是 Trapper 主动连接 Zabbix-Server，所以这里并无实际的意义。

如图 7-6 所示，添加 Items。

- Type: 选择 Zabbix trapper。
- Key: 是自定义的，只需要与 Zabbix-Sender 发送时的 Key 一致即可。

- Type of information: 一定要与 Zabbix-Sender 发送的数据类型一致, 否则, Zabbix-Sender 发送数据会失败。
- Keep history (in days): 历史数据保存的天数。
- Allowed hosts: 只允许哪台机器发送数据给 Zabbix-Server。

The screenshot shows the 'Item' configuration window in Zabbix. The 'Host' field is set to 'trapper'. The 'Name' field is 'trapper'. The 'Type' dropdown is set to 'Zabbix trapper'. The 'Key' field is 'trapperLog', which is circled in red. The 'Type of information' dropdown is set to 'Text'. The 'Keep history (in days)' field is '90'. The 'Allowed hosts' field is empty. The 'New application' field is empty. The 'Applications' dropdown is set to '-None-'. The 'Populates host inventory field' dropdown is set to '-None-'. The 'Description' field is empty. The 'Status' checkbox is checked, indicating 'Enabled'. At the bottom, there are buttons for 'Save', 'Clone', 'Clear history and trends', 'Delete', and 'Cancel'.

图 7-6

7.4.3 使用 zabbix_sender 发送数据

用 zabbix_sender 程序发送数据到 Zabbix-Server, 命令如下:

```
shell# zabbix_sender -z 127.0.0.1 -p 10051 -s "trapper" -k trapper
Log -o "trapper work is ok" -vv
zabbix_sender [21368]: DEBUG: answer [{
    "response": "success",
    "info": "Processed 1 Failed 0 Total 1 Seconds spent 0.000036"
}]
info from server: "Processed 1 Failed 0 Total 1 Seconds spent 0.000036"
sent: 1; skipped: 0; total: 1
```

显示已经发送成功。关于 zabbix_sender 参数, 解释如下。

- -z: Zabbix-Server 的 IP 地址。
- -p: Zabbix-Server 的端口。
- -s: 主机的名称 (确保在添加主机的 Host name, 而不是 visible name)。
- -k: 自定义的 Key。
- -o: 发送的数据。

如果发送失败，会提示失败，需要检测上面的各参数是否正确。

```
shell# zabbix_sender -z 127.0.0.1 -p 10051 -s "trapper" -k trapperLogbad -o "trapper work is ok" -vv
zabbix_sender [23306]: DEBUG: answer [{
    "response": "success",
    "info": "Processed 0 Failed 1 Total 1 Seconds spent 0.000130"
}]
info from server: "Processed 0 Failed 1 Total 1 Seconds spent 0.000130"
sent: 1; skipped: 0; total: 1
```

如图 7-7 所示，在 Latest data 中显示了获取到的数据。

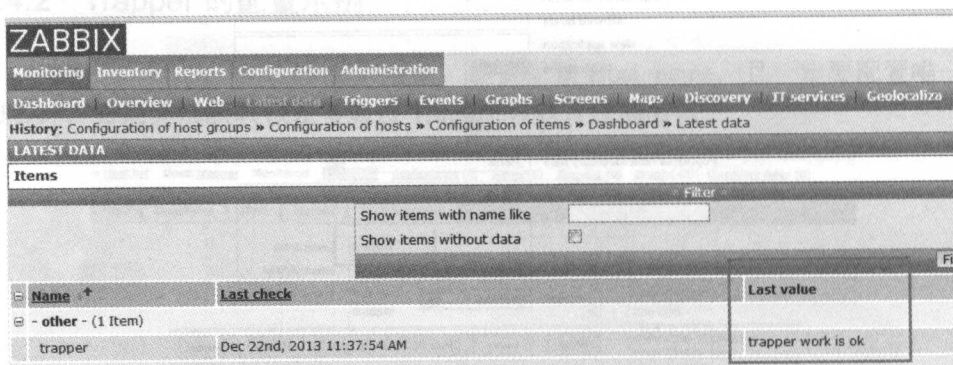


图 7-7

另一点需要注意，就是怎样把数据发送给 Zabbix-Server？前面是用 zabbix_sender 命令手工发送的，如何让其自动上传数据呢？在 Linux 系统中，可以通过定时任务来实现自动发送，或者是第三程序来调用 zabbix_sender 发送数据。

通过上面的学习，我们知道了 zabbix_sender 是一个程序，其程序是由 zabbix 发行版自带的。在实际的环境中，我们完全可以根据需要来写一个，与自己的业务程序相结合使用，具体内容请读者参考 14.2 节。

7.5 SNMP 监控方式

7.5.1 SNMP 概述

SNMP 是 Simple Network Management Protocol 的缩写形式，即简单网络管理协议。简单网络管理包括两部分：管理进程和被管理设备。

管理端和被管理端的通信方式有如下三种。

- 被管理端向管理端发送数据。

- 管理端向被管理端请求获取数据。
- 管理端向被管理端请求改变数据。

基于 TCP/IP 的网络管理包含以下三个组成部分。

- 管理信息库 (management Information Base, MIB): 包含所有代理进程的可被查询和修改的参数。
- 管理信息结构 (Structure of Management Information, SMI): 关于 MIB 的一套公用结构和表示符号。
- 简单网络管理协议 (Simple Network Management Protocol, SNMP): 管理进程和代理进程之间的通信协议。

7.5.2 SNMP 协议的运行

SNMP 协议在 OSI 参考模型的应用层 (第七层) 运行, 支持多种操作, 主要有以下几种基本操作。

- Get 操作: NMS 使用该操作从 Agent 获取一个或多个参数值。
- GetNext 操作: NMS 使用该操作从 Agent 获取一个或多个参数的下一个参数值。
- Set 操作: NMS 使用该操作设置 Agent 一个或多个参数值。
- Response 操作: Agent 返回一个或多个参数值。该操作是前面三种操作的响应。
- Trap 操作: Agent 主动发出的操作, 通知 NMS 有某些事情发生。

SNMP 协议在执行前四种操作时, 使用 UDP 协议, 采用 161 端口发送报文; 执行 Trap 操作时, 设备使用 UDP 协议, 采用 162 端口发送报文。由于收发采用了不同的端口号, 所以一台设备可以同时作为 Agent 和 NMS。

7.5.3 SNMP 协议原理

1. SNMPv1 和 SNMPv2c 实现机制

SNMPv1 和 SNMPv2c 的实现机制基本相同, SNMPv2c 丰富了错误码, 新增了 GetBulk 操作。下面以在 SNMPv1 环境执行 Get、GetNext 和 Set 操作为例, 介绍 SNMPv1 和 SNMPv2c 的实现机制。

(1) Get 操作 (见图 7-8)

NMS 想要获取被管理设备 MIB 节点 sysName 的值 (sysName 对象在允许访问的视图内), 使用 “public” 为可读团体名, 过程如下。

① NMS 给 Agent 发送 Get 请求, 请求报文的主要字段将被设置为: Version 字段的值为 1, Community 字段的值为 public, PDU 的 Variable bindings 中 Name1

字段的值为 sysName.0。

② Agent 给 NMS 发送 Get 响应, 说明是否获取成功。如果成功, 则 Response PDU 的 Variable bindings 中 Value1 字段的值为设备的名字(比如 Agent010-H3C); 如果获取失败, 则在 Error status 字段中填上出错的原因, 在 Error index 中填上出错的位置信息。

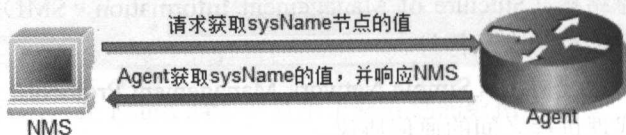


图 7-8

(2) GetNext 操作 (见图 7-9)

NMS 想要获取被管理设备 MIB 节点 sysName 的下一个节点 sysLocation 的值 (sysName 和 sysLocation 对象都在允许访问的视图内), 使用 “public” 为可读团体名, 过程如下:

① NMS 给 Agent 发送 GetNext 请求, 请求报文的主要字段将被设置为: Version 字段的值为 1, Community 字段的值为 public, PDU 的 Variable bindings 中 Name1 字段的值为 sysName.0。

② Agent 给 NMS 发送 GetNext 响应。如果成功, 则 Response PDU 的 Variable bindings 中 Name1 字段的值为 sysName.0 的下一个节点 sysLocation.0, Value1 字段的值为 (比如 Beijing China); 如果获取失败, 则在 Error status 字段中填上出错的原因, 在 Error index 中填上出错的位置信息。

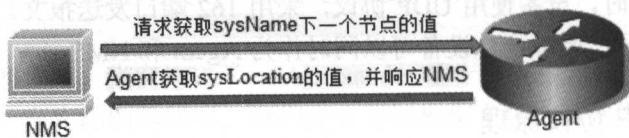


图 7-9

(3) Set 操作 (见图 7-10)

NMS 想要设置被管理设备 MIB 节点 sysName 的值为 Device01, 使用 “private” 为可写团体名, 过程如下。

① NMS 给 Agent 发送 Set 请求, 请求报文的主要字段将被设置为: Version 字段的值为 1, Community 字段的值为 private, PDU 的 Variable bindings 中 Name1 字段的值为 sysName.0, Value1 字段的值为 Device01。

② Agent 给 NMS 发送 Set 响应, 说明是否设置成功。如果设置成功, 则 Response PDU 的 Variable bindings 中 Value1 字段的值为设备的新名字 (比如

Device01); 如果设置失败, 则在 Error status 字段中填上出错的原因, 在 Error index 字段中填上出错的位置信息。

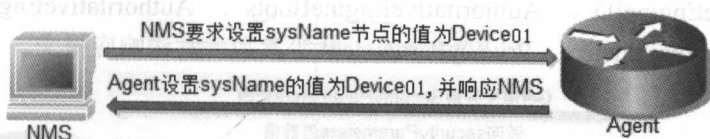


图 7-10

(4) Trap 操作 (见图 7-11)

当设备发生某些异常需要通知 NMS 时, Agent 会主动发出 Trap 报文。例如, 设备某端口网线被拔出, Agent 发送 linkDown 的 Trap 消息给 NMS。Version 字段的值为 1, Community 字段的值为 public, PDU 中 enterprise 字段的取值为 sysObjectID.0 (比如为 enterprises.25506), Generic trap 字段的值为 linkDown, Variable bindings 字段携带接口相关的信息。

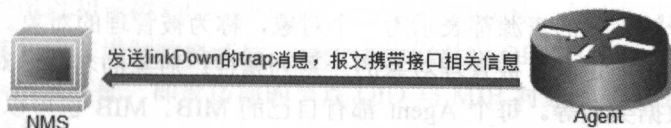


图 7-11

2. SNMPv3 实现机制

SNMPv3 操作的实现机制与 SNMPv1 和 SNMPv2c 基本相同, 其主要区别在于 SNMPv3 新增加了认证、加密和解密的处理。下面以 SNMPv3 使用认证和加密方式执行 Get 操作为例, 描述其实现机制, 如图 7-12 所示, 过程如下。

① NMS 首先发送不带任何认证和加密参数的 Get 请求, Flags 字段设置为 0x4, 以获取 contextEngineID、contextName、AuthoritativeEngineID、AuthoritativeEngineBoots、AuthoritativeEngineTime 等相关参数的值。

② Agent 解析消息, 发送 Report 报文, 并携带上述相关参数的值。

③ NMS 再次给 Agent 发送 Get 请求, 请求报文的主要字段将被设置为: Version 字段的值为 3, 将步骤②获取到的参数值填入相应的字段, PDU 的 Variable bindings 中 Name1 字段的值为 sysName.0, 并且根据配置的认证算法计算出 AuthenticationParameters, 使用配置的加密算法计算出 PrivacyParameters, 并使用配置的加密算法对 PDU 数据进行加密。

④ Agent 首先对消息进行认证, 认证通过后对 PDU 报文进行解密。解密成功后, 则获取 sysName.0 对象的值, 并将 Response PDU 的 Variable bindings 中 Value1 字段的值填为设备的名字 (比如 Agent010)。如果认证、解密失败或者获

取参数值失败，则在 *Error status* 字段中填上出错的原因，在 *Error index* 字段中填上出错的位置信息。最后对 PDU 进行加密，设置 *contextEngineID*、*contextName*、*AuthoritativeEngineID*、*AuthoritativeEngineBoots*、*AuthoritativeEngineTime*、*AuthenticationParameters*、*PrivacyParameters* 等参数，发送响应报文。

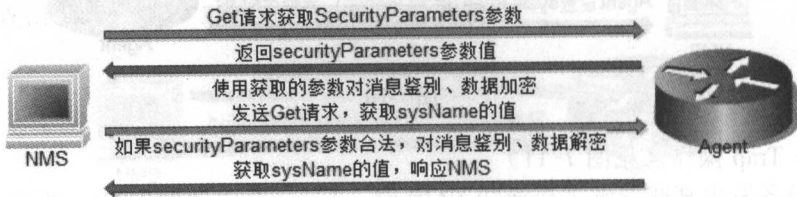


图 7-12

7.5.4 MIB 简介

1. MIB 的含义

任何一个被管理的资源都表示为一个对象，称为被管理的对象。MIB 是被管理对象的集合，它定义了被管理对象的一系列属性：对象的名称、对象的访问权限和对象的数据类型等。每个 Agent 都有自己的 MIB。MIB 也可以看作是 NMS 和 Agent 之间的一个接口，通过这个接口，NMS 可以对 Agent 中的每一个被管理对象进行读/写操作，从而达到管理和监控设备的目的。NMS、Agent 和 MIB 之间的关系如图 7-13 所示。

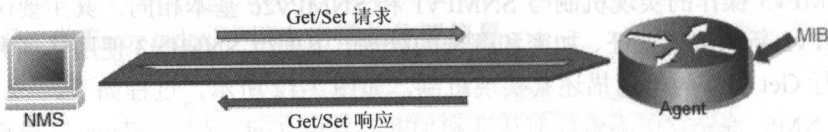


图 7-13

2. MIB 视图

MIB 视图是 MIB 的子集合，配置 Agent 时，用户可以将团体名/用户名与 MIB 视图绑定，从而限制 NMS 能够访问的 MIB 对象。用户可以配置 MIB 视图内的对象为 *excluded* 或 *included*。*excluded* 表示当前视图不包括该 MIB 子树的所有节点；*included* 表示当前视图包括该 MIB 子树的所有节点。

3. OID 和子树

MIB 是以树状结构进行存储的。树的节点表示被管理对象，它可以用从根开始的一条路径唯一地识别，这条路径就称为 *OID*。如图 7-14 所示，管理对象 *system* 可以用一串数字 {1.3.6.1.2.1.1} 唯一标识，这串数字就是 *system* 的 *OID*。

子树可以用该子树根节点的 OID 来标识。如以 private 为根节点的子树的 OID 为 private 的 OID——{1.3.6.1.4}。

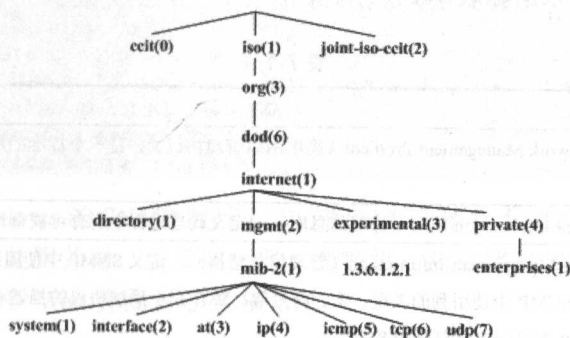


图 7-14

4. 子树掩码

子树掩码可以和子树 OID 共同确定一个视图的范围。子树掩码用十六进制数格式表示，转化成二进制数格式后，每个比特位对应 OID 中的一个小节，其中：

- 1 表示精确匹配，即要访问的节点 OID 与 MIB 对象子树 OID 对应小节的值必须相等。
- 0 表示通配，即要访问的节点 OID 与 MIB 对象子树 OID 对应小节的值可以不相等。

例如，子树掩码为 0xDB (二进制数为 11011011)，子树 OID 为 1.3.6.1.6.1.2.1，则对应关系如图 7-15 所示，所确定的视图就包括子树 OID 为 1.3.*.1.6.*.2.1 (*表示可为任意数字) 子树下的所有节点。

子树OID	1	3	6	1	6	1	2	1
子树掩码	1	1	0	1	1	0	1	1

图 7-15

说明：

- 若子树掩码的位数目大于子树 OID 的小节数，则匹配时，子树掩码的第一位与子树 OID 的第一小节对齐，第二位与第二小节对齐，依此类推，子树掩码中多出的位将被忽略。
- 若子树掩码的位数目小于子树 OID 的小节数，则匹配时，子树掩码的第一位与子树 OID 的第一小节对齐，第二位与第二小节对齐，依此类推，子树掩码中不足的位将自动设置为 1。
- 如果没有指定子树掩码，则使用默认的子树掩码 (全 1)。

7.5.5 SNMP 的相关术语

SNMP 的相关术语如表 7-1 所示。

表 7-1

名 称	描 述
SNMP	Simple Network Management Protocol（简单网络管理协议），是一个标准的用于管理基于 IP 网络设备的协议
MIB	Management Information Base（管理信息库），定义代理进程中所有可被查询和修改的参数
SMI	Structure of Management Information（管理信息结构），定义 SNMP 中使用到的 ASN.1 类型和语法，并定义 SNMP 中使用到的类型、宏、符号等。SMI 用于后续协议的描述和 MIB 的定义。每个版本的 SNMP 都可能定义自己的 SMI
ASN.1	Abstract Syntax Notation One（抽象语法定义），用于定义语法的正式语言，在 SNMP 中定义 SNMP 的协议数据单元 PDU 和管理对象 MIB 的格式。SNMP 只使用了 ASN.1 中的一部分，而且使用 ASN.1 的语言特性定义了一些自定义类型和类型宏，这些组成了 SMI
PDU	Protocol Data Unit（协议数据单元），它是网络中传送的数据包。每一种 SNMP 操作在物理上都对应一个 PDU
NMS	Network Management System（网络管理系统，又名网络管理站，简称“管理站”），是 SNMP 的总控机，提供统一的用户界面访问支持 SNMP 的设备，一般提供 UI 界面，并有统计、分析等功能，是网管系统的总控制台。NMS 是网络管理操作的发起者
Agent	是 SNMP 的访问代理，简称“代理”，为设备提供 SNMP 能力，负责设备与 NMS 的通信
Proxy	代理服务器，对实现不同协议的设备进行协议转换，使非 IP 协议的设备也能被管理
Trap	是由设备主动发出的告警数据，用于提示重要状态的改变
BER	Basic Encoding Rule（基本编码规格），描述如何将 ASN.1 类型的值编码为字符串的方法。它是 ASN.1 标准的一部分。BER 编码将数据分成 TLV 三部分：T 为 Tag 的缩写，是类型标识；L 为 Length 的缩写，是标识类型的长度；V 为 Value 的缩写，是标识数据内容。按照 TLV 的顺序对数据进行编码，生成字节流。SNMP 使用 BER 将 SNMP 的操作请求和应答编码后进行传输，并用于接收端进行解码

7.5.6 配置 Zabbix 以 SNMP 方式监控

使用 SNMP 可以监控路由器、交换机、打印机、UPS 或者是其他开启 SNMP 的设备，如果要支持 SNMP 的监控方式，需要 Zabbix-Server 在源码编译的时候带上 with-net-snmp 参数，语句如下。

```
shell ./configure --with-net-snmp
```

1. 配置被监控端的 SNMP

以 Linux 为例，语句如下。

```

shell# yum -y install net-snmp
shell# mv /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.bak
shell# vim /etc/snmpd/snmpd.conf
com2sec mynetwork 192.168.0.240 public_monitor
com2sec mynetwork 127.0.0.1 public
group MyROGroup v2c mynetwork
access MyROGroup "" any noauth prefix all none none
view all included .1 80
shell# chkconfig snmpd on
shell# service snmpd restart

```

如果是 Windows 的 SNMP 监控方式，配置方法稍有不同。

如果设备不是服务器，而是路由器、交换机、防火墙等其他硬件设备，则需要通过命令行或者 Web 界面去配置 SNMP。

2. 测试能否获取 SNMP 数据

在 Zabbix-Server 上测试，语句如下。

```

shell# snmpwalk -v 2c -c public 127.0.0.1
shell# snmpwalk -v 2c -c public 127.0.0.1 SNMPv2-MIB::sysUpTime.0
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (166696) 0:27:46.96

```

如果没有 snmpwalk 命令，请安装 net-snmp-utils (类 RHEL 系统)，语句如下。

```
shell# yum install net-snmp-utils
```

3. 添加 SNMP 的 Items

添加 Items 时，可以在 Host 中，也可以在 Template 中完成，如图 7-16 所示。

Template list Template: Template SNMP Generic Applications (1) Items (5) Triggers (0) Graphs (0)

cmi

Name: Device name

Type: SNMPv2 agent

Key: sysName Select

SNMP OID: SNMPv2-MIB::sysName.0

SNMP community: {\$SNMP_COMMUNITY}

Port:

Type of information: Character

Update interval (in sec): 3600

Flexible intervals

Interval	Period	Action
No flexible intervals defined.		

New flexible interval Interval (in sec): 50 Period: 1-7,00:00-24:00 Add

Keep history (in days): 7

Show value: As is show value mappings

New application:

Applications: -None-
General

图 7-16

4. Items SNMP 各参数说明

图 7-16 中各参数的说明如表 7-2 所示。

表 7-2

名 称	作 用
Name	Items 的名称
Type	选择 Type 为 SNMPv2 agent
Key	必须为一个唯一的字符串，这个 Key 是后面配置触发器时要用到的
SNMP OID	填写 OID 的值
SNMP community	community 的值，默认是一个变量（macro），可以设置为一个实际的参数或值
Port	SNMP 的端口

5. 配置 SNMP community 的 Macro

前面我们配置的 SNMP community 是 public_monitor，而默认的 community 值为{\$SNMP_COMMUNITY}，在全局变量中，{\$SNMP_COMMUNITY}的值为 public，所以需要对其{\$SNMP_COMMUNITY}的变量进行重新定义。

在 Template 中定义 Macros，如图 7-17 所示。

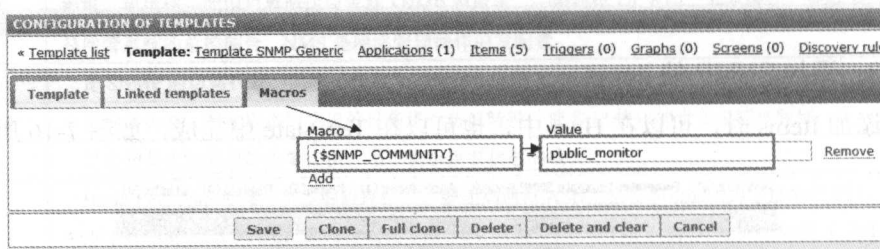


图 7-17

在 Host 中定义 Macros，如图 7-18 所示。

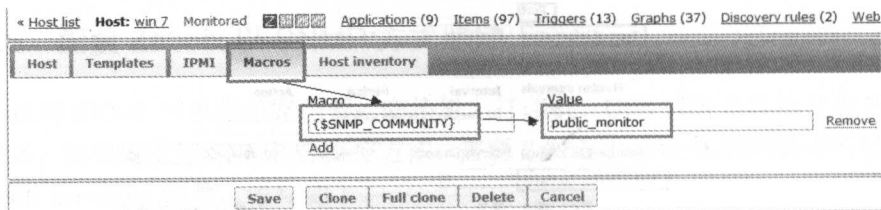


图 7-18

配置 Items 后，可以按需配置 Graphs 和 Trigger 等，其方法与 Agent 方式相同。

7.6 IPMI 监控方式

1. IPMI 的概念

IPMI (Intelligent Platform Management Interface) 即智能平台管理接口, 原本是 Intel 架构中企业系统的周边设备所采用的一种工业标准, 后来成为业界通用的标准。

用户可以利用 IPMI 监视服务器的物理特征, 如温度、电压、电扇工作状态、电源供应以及机箱入侵等。

2. IPMI 的特性

IPMI 独立于 CPU BIOS 和 OS 外自行运行, 允许管理者在缺少操作系统、系统管理软件或受监控的系统关机但有接电源的情况下仍能远端管理服务器硬件。IPMI 也能在操作系统启动后活动, 与系统管理功能一并使用时还能提供加强功能, IPMI 只定义架构和接口格式成为标准, 具体操作时可能会有所不同。

关于 IPMI 的更多信息, 请读者自行参考相关书籍。

3. 配置 Zabbix-Server 监控 IPMI

步骤如下。

① 若要支持 IPMI, 必须在安装并编译 `zabbix_server` 程序时带上 `--with-openipmi` 编译参数。

② 服务器端配置 `StartIPMIPollers` 参数, 默认参数值为 0, 需修改为大于 0 的参数, 即可开启对 IPMI 的支持, 如这里将改为 5, 表示启动 IPMI 线程为 5 个。

```
shell# sed -i '/# StartIPMIPollers=0/aStartIPMIPollers=5' zabbix_server.conf
shell# service zabbix-server restart
```

在 Zabbix 中, 对 IPMI 的支持是通过获取 IPMI 中 sensor 的参数获取数据。Zabbix-Server 在获取 IPMI 监控数据的时候, `zabbix_server.conf` 开启 `DebugLevel=4`, 会添加 Added sensor 字符的日志。程序代码在源码的 `src/zabbix_server/poller/checks_ipmi.c` 中, 如下:

```
zabbix_log(LOG_LEVEL_DEBUG, "Added sensor: host:'%s:%d' id_type:%d id_sz:%d id:'%s'"
    " reading_type:0x%x ('%s') type:0x%x ('%s') full_name:'%s'",
    h->ip, h->port,
    s->id_type, s->id_sz, sensor_id_to_str(id_str, sizeof(id_str), s->id, s->id_type, s->id_sz),
    s->reading_type, ipmi_sensor_get_event_reading_type_string(s->sensor), s->type,
    ipmi_sensor_get_sensor_type_string(s->sensor), full_name);
```


Zabbix 中 IPMI 的设计文档网址如下。

<https://www.zabbix.org/wiki/Docs/specs/ZBXNEXT-300>

4. Zabbix 自带的 IPMI 模板

Zabbix 默认的 IPMI 模板如图 7-19 所示。

<input type="checkbox"/> Template IPMI Intel SR1530	Applications (3)	Items (8)	Triggers (11)	Graphs (2)	Screens (0)	Discovery (0)	Web (0)
<input type="checkbox"/> Template IPMI Intel SR1630	Applications (3)	Items (11)	Triggers (21)	Graphs (2)	Screens (0)	Discovery (0)	Web (0)

图 7-19

Key 的设置如图 7-20 所示。

默认的 IPMI 模板中 Key 值有限,在某些硬件中,还有其他更多的 IPMI 参数。

Template list Template: Template IPMI Intel SR1530 Applications (3) Items (8) Triggers (11) Graphs (2) S					
<input type="checkbox"/> Wizard	Name *	Triggers	Key	Interval	
<input type="checkbox"/>	BB +1.8V SM	Triggers (2)	bb_1.8v_sm	60	
<input type="checkbox"/>	BB +3.3V	Triggers (2)	bb_3.3v	60	
<input type="checkbox"/>	BB +3.3V STBY	Triggers (2)	bb_3.3v_stby	60	
<input type="checkbox"/>	BB +5.0V	Triggers (2)	bb_5.0v	60	
<input type="checkbox"/>	BB Ambient Temp	Triggers (2)	bb_ambient_temp	60	
<input type="checkbox"/>	Power	Triggers (1)	power	60	
<input type="checkbox"/>	Processor Vcc		processor_vcc	60	
<input type="checkbox"/>	System Fan 3		system_fan_3	60	

图 7-20

5. 在 Linux 系统中使用 OpenIPMI

```
shell# yum install OpenIPMI ipmitool
shell# service ipmi start
```

(1) 启动 IPMI 服务

```
shell# /etc/init.d/ipmi start
starting ipmi drivers: [ ok ]
shell# /etc/init.d/ipmievdd start
starting ipmievdd: [ ok ]
```

注意: 必须要 ipmi、ipmievdd 服务启动成功后才能运行 ipmitool 命令。

(2) 配置 IPMI 地址

配置 IPMI 地址

```
shell# ipmitool lan print 1    #显示lan 1的配置信息
shell# ipmitool lan set 1 ipaddr 10.10.10.10
shell# ipmitool lan set 1 netmask 255.255.255.0
shell# ipmitool lan set 1 defgw ipaddr 10.10.10.1
```

配置用户

```
shell# ipmitool lan set 1 access on    #开启lan 1的用户访问
shell# ipmitool user list 1    #列出lan 1的用户
```

```
shell# ipmitool user set name 10 sensor
shell# ipmitool user set password 10 sensor
shell# ipmitool user enable 10
shell# ipmitool user priv 10 2 1
shell# ipmitool user list 1
```

ID	Name	Callin	Link	Auth	IPMI Msg	Channel	Priv	Limit
2	root	true	true	true		ADMINISTRATOR		
10	sensor	true	false	true		USER		

(3) ipmitool 常用命令

- shell# ipmitool -I lan -H 服务器地址 -U root -P 密码 power off (硬关机, 直接切断电源)。
- shell# ipmitool -I lan -H 服务器地址 -U root -P 密码 power soft (软关机, 即如同轻按一下开机按钮)。
- shell# ipmitool -I lan -H 服务器地址 -U root -P 密码 power on (硬开机)。
- shell# ipmitool -I lan -H 服务器地址 -U root -P 密码 power reset (硬重启, 这也许会常用)。
- shell# ipmitool -I lan -H 服务器地址 -U root -P 密码 power status (获取当前电源状态)。

(4) 查看 IPMI 支持的参数

```
shell# ipmitool -H 10.10.10.10 -U sensor -L USER sensor list
```

如图 7-21、图 7-22 所示, 查看 IPMI 所能获取到的数据值。

[root@ ~]# ipmitool sensor									
UID Light	0x0	discrete	0x0080	na	na	na	na	na	na
Health LED	0x0	discrete	0x0080	na	na	na	na	na	na
VRM 1	0x0	discrete	0x0280	na	na	na	na	na	na
VRM 2	0x0	discrete	0x0280	na	na	na	na	na	na
VRM 3	0x0	discrete	0x0280	na	na	na	na	na	na
VRM 4	0x0	discrete	0x0280	na	na	na	na	na	na
Inlet Ambient	17.000	degrees C	ok	na	na	na	na	42.000	46.000
System board	24.000	degrees C	ok	na	na	na	na	85.000	90.000
CPU 1	20.000	degrees C	ok	na	na	na	na	75.000	80.000
CPU 2	18.000	degrees C	ok	na	na	na	na	75.000	80.000
CPU 3	16.000	degrees C	ok	na	na	na	na	75.000	80.000
CPU 4	16.000	degrees C	ok	na	na	na	na	75.000	80.000
DIMMs 1	38.000	degrees C	ok	na	na	na	na	87.000	92.000
DIMMs 2	32.000	degrees C	ok	na	na	na	na	87.000	92.000
DIMMs 3	27.000	degrees C	ok	na	na	na	na	87.000	92.000
DIMMs 4	34.000	degrees C	ok	na	na	na	na	87.000	92.000
Mem 1 1-4 Zone	38.000	degrees C	ok	na	na	na	na	85.000	90.000
Mem 1 5-8 Zone	38.000	degrees C	ok	na	na	na	na	85.000	90.000
Mem 2 1-4 Zone	30.000	degrees C	ok	na	na	na	na	85.000	90.000
Mem 2 5-8 Zone	32.000	degrees C	ok	na	na	na	na	85.000	90.000
Mem 3 1-4 Zone	24.000	degrees C	ok	na	na	na	na	85.000	90.000
Mem 3 5-8 Zone	25.000	degrees C	ok	na	na	na	na	85.000	90.000
Mem 4 1-4 Zone	28.000	degrees C	ok	na	na	na	na	85.000	90.000
Mem 4 5-8 Zone	28.000	degrees C	ok	na	na	na	na	85.000	90.000
IOH 1	57.000	degrees C	ok	na	na	na	na	100.000	105.000
IOH 2	49.000	degrees C	ok	na	na	na	na	100.000	105.000
NIC Zone	58.000	degrees C	ok	na	na	na	na	90.000	95.000
Mezz Zone	52.000	degrees C	ok	na	na	na	na	90.000	95.000
Chassis Exit	57.000	degrees C	ok	na	na	na	na	100.000	105.000
HDD Max	35.000	degrees C	ok	na	na	na	na	60.000	65.000
Prochot	0x0	discrete	0x0280	na	na	na	na	na	na
Virtual Fan	27.440	unspecified	nc	na	na	na	na	na	na
Enclosure Status	0x0	discrete	0x0080	na	na	na	na	na	na
Power Meter	274.000	watts	cr	na	na	na	na	na	na
Memory Status	0x0	discrete	0x4080	na	na	na	na	na	na
Cntlr 1 Bay 1	0.000	unspecified	nc	na	na	na	na	na	na
Cntlr 1 Bay 2	0.000	unspecified	nc	na	na	na	na	na	na
Cntlr 1 Bay 3	0.000	unspecified	ok	na	na	na	na	na	na
Cntlr 1 Bay 4	0.000	unspecified	ok	na	na	na	na	na	na

图 7-21

6. 创建 IPMI 模板

若要创建 VRM 1 的监控, 则应添加如下 Key。

Name: 可以随意取一个有意义的名称。

Type: 选择 IPMI agent。

Key: 填充为 VRM_1（注意，VRM_1 为 Sensor ID 的 VRM 1），如图 7-23 所示，是 IPMI 可以获取到的数据。

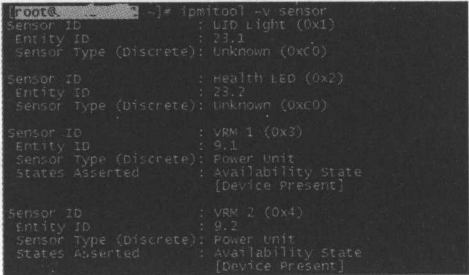


图 7-22

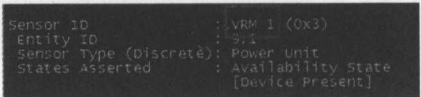


图 7-23

本例为 HP G7 模板的配置，创建 Template IPMI HP G7 模板，例如，创建 VRM_1 的 Items，如图 7-24 所示。

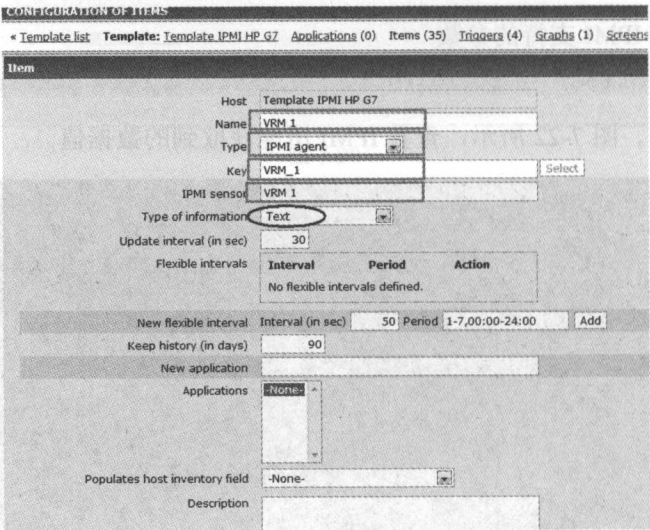


图 7-24

设置触发器，如图 7-25 至图 7-29 所示。

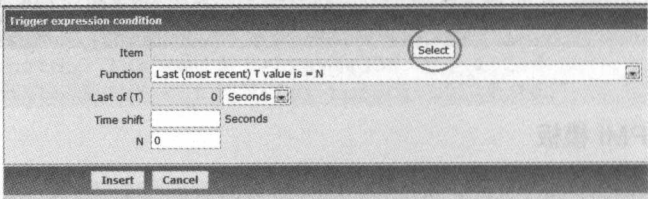


图 7-25

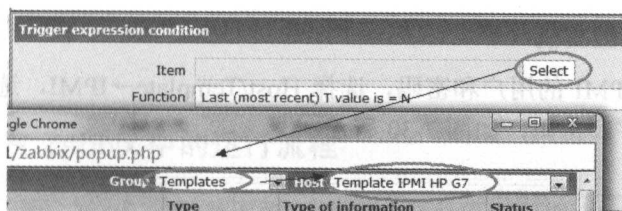


图 7-26

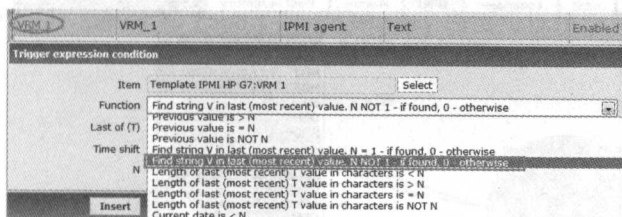


图 7-27

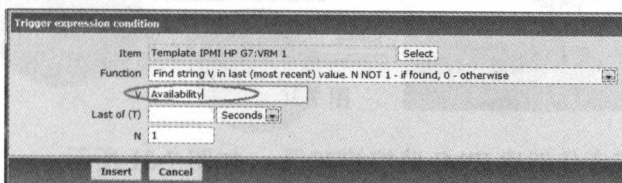


图 7-28

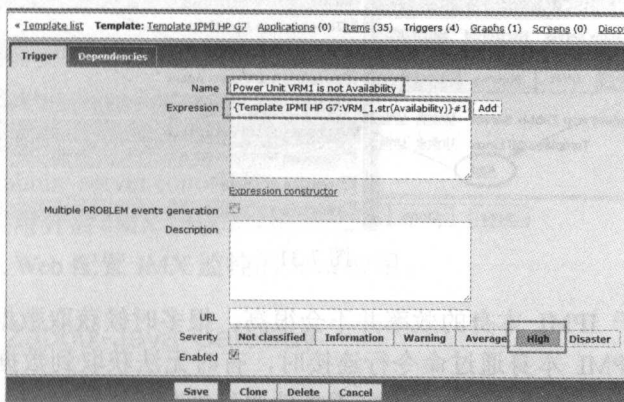


图 7-29

如果结果中有 Availability 字符串，则结果为 1，没有找到则不为 1，即触发告警规则。

其他的 Items，依次添加即可。模板位于 <https://github.com/itnihao/zabbix-book/tree/master/07-chapter> 中。

7. IPMI 监控主机

若要配置 IPMI 的用户和密码，选择 Host/Template→IPMI，选择认证方式、用户类型、名字、密码，如图 7-30 所示。

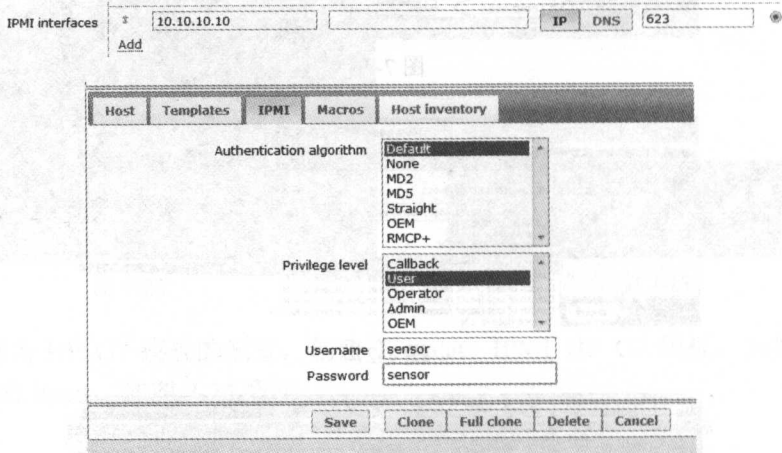


图 7-30

选择模板或者是创建 IPMI 监控的 Items，如图 7-31 所示。

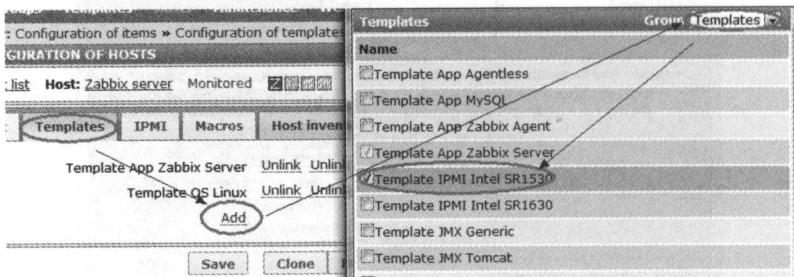


图 7-31

说明：由于 IPMI 本身的效率并不会很高，很多时候获取数据会不连续。经过笔者测试，IPMI 本身通过命令行连接时，有时无法获取到数据，所以要监控 IPMI，在设置触发器时，需要利用多重条件进行判断，防止误报。另外，需要加长检测周期，建议至少在 5~10 分钟检测一次。

7.7 JMX 监控方式

JMX（Java Management Extensions，即 Java 管理扩展）是 Java 平台上为应用程序、设备、系统等植入管理功能的框架。JMX 可以跨越一系列异构操作系统平

台、系统体系结构和网络传输协议，灵活地开发无缝集成的系统、网络和服务管理应用。

7.7.1 JMX 在 Zabbix 中的运行流程

在 Zabbix 中，JMX 监控数据的获取由专门的代理程序来实现，即 Zabbix-Java-Gateway 来负责数据的采集，Zabbix-Java-Gateway 和 JMX 的 Java 程序之间通信获取数据，Zabbix-Server 和 Zabbix-Java-Gateway 通信可以用图 7-32 来表示。

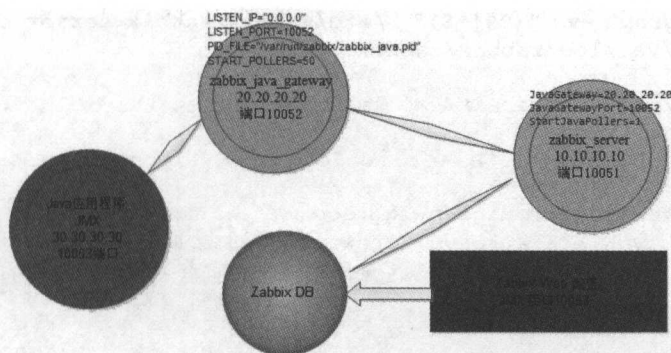


图 7-32

7.7.2 配置 JMX 监控的步骤

JMX 监控的安装和配置步骤如下。

- ① 安装 Zabbix-Java-Gateway。
- ② 配置 zabbix_java_gateway.conf 参数。
- ③ 配置 zabbix_server.conf 参数。
- ④ Java 应用开启 JMX 协议。
- ⑤ Zabbix Web 配置 JMX 监控的 Java 应用。

7.7.3 安装 Zabbix-Java-Gateway

源码安装的时候，加--enable-java 参数，依赖关系有 java、java-devel 软件包。

RPM 安装方式（注意不能与 Zabbix-Server 安装在同一台机器中）的语句如下：

```
shell# yum install java java-devel zabbix-java-gateway
```

源码安装方式的语句如下：（注意修改启动脚本，配置文件，可以参考 RPM 安装的配置文件和脚本。）

```
shell# ./configure --enable-java --prefix=/usr
```


7.7.4 配置 Zabbix-Java-Gateway

Zabbix-Java-Gateway 的配置语句如下。

```
shell# egrep '=' /etc/zabbix/zabbix_java_gateway.conf
LISTEN_IP="0.0.0.0"
LISTEN_PORT=10052
PID_FILE="/var/run/zabbix/zabbix_java.pid"
START_POLLERS=50
```

以下是启动 50 个进程的语句。

```
shell# egrep -v "(^#|^$)" /etc/zabbix/zabbix_server.conf
LogFile=/var/log/zabbix/zabbix_server.log
LogFileSize=0
PidFile=/var/run/zabbix/zabbix_server.pid
DBName=zabbix
DBUser=zabbix
DBPassword=zabbix
DBSocket=/var/lib/mysql/mysql.sock
JavaGateway=X.X.X.X #Java-Gateway服务器的IP地址，如果Java-Gateway和
                      #Zabbix-Server在一台机器中，就可以写为127.0.0.1
JavaGatewayPort=10052
StartJavaPollers=5
ExternalScripts=/etc/zabbix/externalscripts
```

Zabbix-Server 中的参数 StartJavaPollers 和 Zabbix-Java-Gateway 的参数 START_POLLERS，需要注意的语句如下。

```
StartJavaPollers <= START_POLLERS
```

如果不满足以上条件，就会出现 Zabbix-Server 向 Zabbix-Java-Gateway 发出请求后无响应的情况。表 7-3 说明了 Zabbix-Java-Gateway 和 Zabbix-Server 需要配置的参数。

表 7-3

Zabbix-Java-Gateway	Zabbix-Server	注 意 事 项
LISTEN_IP="0.0.0.0"	JavaGateway=X.X.X.X	X.X.X.X 参数为 Zabbix-Java-Gateway 的 IP
LISTEN_PORT=10052	JavaGatewayPort=10052	端口必须一致，且防火墙允许
START_POLLERS=50	StartJavaPollers=5	StartJavaPollers <= START_POLLERS

7.7.5 监控 Java 应用程序

假设 Java 程序为/usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar，若想监控该程序的运行情况，那么为了开启 JMX 的支持，用以下命令启动这个 Java 程序。

```

shell# java \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=10053 \
-Dcom.sun.management.jmxremote.authenticate=false \
-Dcom.sun.management.jmxremote.ssl=false \
-jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar

```

如果不需要 JMX，则启动命令如下：

```

shell# java -jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar

```

如果采用认证加密的方式，语句如下：

```

shell# java \
-Djava.rmi.server.hostname=192.168.0.200\
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=10053 \
-Dcom.sun.management.jmxremote.authenticate=true \
-Dcom.sun.management.jmxremote.password.file=/etc/java-6-openjdk/management/jmxremote.password \
-Dcom.sun.management.jmxremote.access.file=/etc/java-6-openjdk/management/jmxremote.access \
-Dcom.sun.management.jmxremote.ssl=true \
-Djavax.net.ssl.keyStore=$YOUR_KEY_STORE \
-Djavax.net.ssl.keyStorePassword=$YOUR_KEY_STORE_PASSWORD \
-Djavax.net.ssl.trustStore=$YOUR_TRUST_STORE \
-Djavax.net.ssl.trustStorePassword=$YOUR_TRUST_STORE_PASSWORD \
-Dcom.sun.management.jmxremote.ssl.need.client.auth=true \
-jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar

```

如果是 Tomcat、Weblogic 程序，则应把上面这些信息添加到启动文件。

7.7.6 自定义 JMX 的 Key

1. 通过 jconsole 获取数据

在 Windows 系统中安装 JDK，找到 jconsole.exe，如图 7-33 所示。

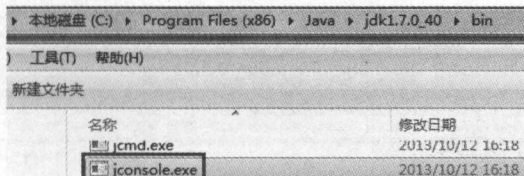


图 7-33

连接 JMX 服务，这里为 10.10.10.10:10053，表示匿名用户连接，如图 7-34 所示。

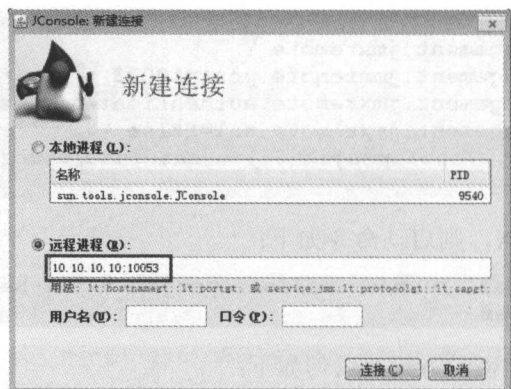


图 7-34

例如，参数 Memory 下的 NonHeapMemoryUsage，其设置如图 7-35 所示。

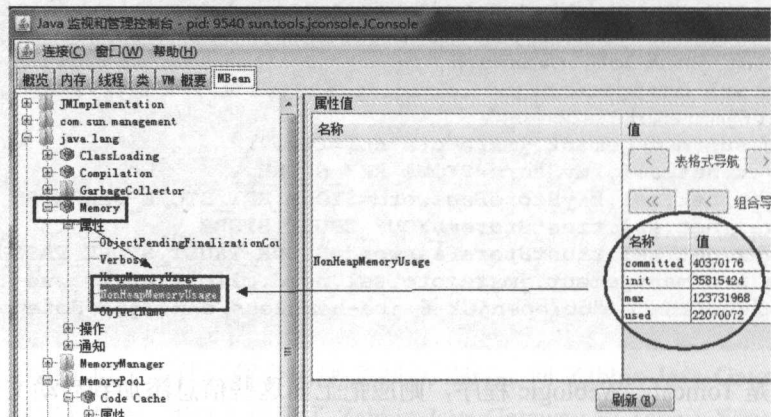


图 7-35

2. 检测 JMX 数据的获取

通过命令行获取 jconsole 中的数据。

```
shell# wget http://crawler.archive.org/cmdline-jmxclient/cmdline-jmxclient-0.10.3.jar
shell# java -jar cmdline-jmxclient-0.10.3.jar - 10.10.10.10:10053 java.lang:type=Memory NonHeapMemoryUsage
10/25/2013 18:07:49 +0800 org.archive.jmx.Client NonHeapMemoryUsage:
committed: 32178176
init: 24313856
max: 224395246
used: 15336752
```

通过上面的命令，我们可以对 JMX 数据进行查询，这样就能验证 JMX 配置和监控的值是否正确，同时也能够发现 Key 需要的参数。

3. 配置 JMX 的 Items

在 Items 中的设置如图 7-36 所示, Key 来自图 7-35 中的数据, 在 Items 添加语句如下:

```
jmx["java.lang:type=MemoryPool",NonHeapMemoryUsage.used]
```

Template: Template JMX Generic Applications (8) Items (55) Triggers (26) Graphs (11) Screens (1)

Name: mem Non-Heap Memory used

Type: JMX agent

Key: jmx["java.lang:type=Memory",NonHeapMemoryUsage.used]

User name:

Password:

Type of information: Numeric (unsigned)

Data type: Decimal

图 7-36

相信通过这个例子的学习, 读者应该知道如何自定义 JMX 的 Items。

7.7.7 监控 Tomcat

1. Tomcat 配置 JMX

```
shell# vim /usr/sbin/tomcat6 #源码安装修改catalina.sh, 放在开头即可。
export CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote"
export CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote.port=10053"
export CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote.authenticate=false"
export CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote.ssl=false"
```

将以上内容添加到如图 7-37 框线所示的位置。

```
# CLASSPATH munging
if [ -n "$JSE_HOME" ]; then
    CLASSPATH="${CLASSPATH}:${build-classpath jcert jnet jsse 2>/dev/null}"
fi
CLASSPATH="${CLASSPATH}:${CATALINA_HOME}/bin/bootstrap.jar"
CLASSPATH="${CLASSPATH}:${CATALINA_HOME}/bin/tomcat-juli.jar"
CLASSPATH="${CLASSPATH}:${build-classpath commons-daemon 2>/dev/null}"

export CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote"
export CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote.port=10053"
export CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote.authenticate=false"
export CATALINA_OPTS="$CATALINA_OPTS -Dcom.sun.management.jmxremote.ssl=false"

if [ "$1" = "start" ]; then
```

图 7-37

重启 Tomcat 程序的语句如下。

```
shell# service tomcat6 stop
shell# service tomcat6 start
```

2. Web 中对主机添加 JMX 监控方式

添加 Host 的时候, 选择 JMX 的监控方式, 端口值为 10053, 这里的 IP 地址 10.1.1.5 (被监控端) 是 JMX 监控方式所在的主机, 如图 7-38 所示。



图 7-38

选择 Tomcat 模板, 如图 7-39 所示。

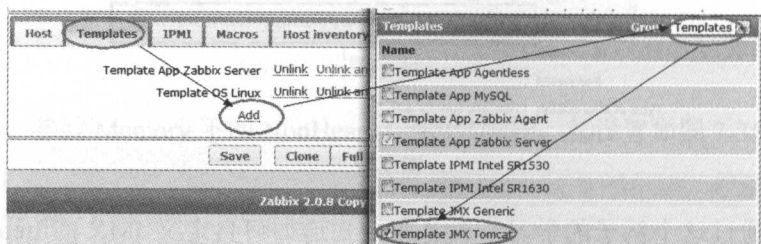


图 7-39

7.7.8 Weblogic 的监控

首先安装 Weblogic, 本例的安装目录为 /opt/weblogic, 实例为 webapp。

```
shell# cd /opt/weblogic/user_projects/domains/webapp/
shell# vim bin/setDomainEnv.sh
export JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote"
export JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote.port=10053"
export JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote.authenticate=false"
export JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote.ssl=false" #修改的参数位置如图7-40框线所示部分
```

```
JAVA_PROPERTIES="${JAVA_PROPERTIES} ${CLUSTER_PROPERTIES}"
export JAVA_PROPERTIES

JAVA_DEBUG=""
export JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote"
export JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote.port=10053"
export JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote.authenticate=false"
export JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote.ssl=false"

if [ "${debugFlag}" = "true" ]; then
    JAVA_DEBUG="-Xdebug -Xnoagent -Xrunjdp:transport=dt_socket,address=${DEBUG_PORT},ser
    export JAVA_DEBUG
    JAVA_OPTIONS="${JAVA_OPTIONS} ${enableHostswapFlag} -ea -da:com.bea... -da:javelin...
    sole..."
    export JAVA_OPTIONS
fi
```

图 7-40

shell# ./startWebLogic.sh #启动WebLogic服务, 如图7-41所示

用 ps 命令查看启动的进程, 可查看刚才添加的 JMX 相关参数, 如图 7-42 所示。


```
[root@web-db bin]# ./startweblogic.sh
JAVA Memory arguments: -Xms256m -Xmx512m -XX:MaxPermSize=128m
WLS Start Mode=Production
CLASSPATH=/opt/weblogic/user_projects/domains/sicloudomain/lib/antlc-2.7.6.jar:/
1036/profiles/default/sys_manifest_classpath/weblogic_patch.jar:/opt/weblogic/pa
/lib/tools.jar:/opt/weblogic/wlserver_10.3/server/lib/weblogic_sp.jar:/opt/web
er/modules_10.3.6.0.jar:/opt/weblogic/wlserver_10.3/server/lib/webservices.jar:/
antcontrib_1.1.0.0-1.0b2/lib/ant-contrib.jar:/opt/weblogic/wlserver_10.3/common/
PATH=/opt/weblogic/wlserver_10.3/server/bin:/opt/weblogic/modules/org.apache.ant
/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/jdk1.7/bin:/usr/local/jdk1.7/jre/b
*****
* To start WebLogic Server, use a username and *
* password assigned to an admin-level user. For *
* server administration, use the WebLogic Server *
* console at http://hostname:port/console *
*****
starting weblogic with Java version:
java version "1.7.0_45"
Java(TM) SE Runtime Environment (build 1.7.0_45-b15)
Java HotSpot(TM) 64-bit Server VM (build 24.45-b08, mixed mode)
```

图 7-41

```
mixed mode)
512m -XX:MaxPermSize=128m -Dweblogic.Name=AdminServer -Dweblogic.SecurityPolicy=/opt/weblogic/10.3.6.0/lib
true -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=10053 -Dcom.sun.management.jmxremote.authen
is -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.ssl.enabled=false -Dcom.sun.management.jmxremote
ment.discover=true -Dwls.iterativebeve=false -Dwls.testconsole=false -Dwls.logerrorsToconsole=false -Dweblogic.ext
/sect_manifest_classpath:/opt/weblogic/patch_ocp371/profiles/default/sysext_manifest_classpath weblogic.Server
EA-090905> <Disabling CryptoJ JCE Provider self-integrity check for better startup performance. To enable this che
Verification=true>
```

图 7-42

添加监控模板的内容在这里就不详叙了。

7.8 命令的执行

1. system.run

system.run 是 Zabbix-Agent 的一个 Key, 这个 Key 在默认情况下是不能使用的。若要使用这个 Key, 需在 /etc/zabbix/zabbix_agentd.conf 中修改为 EnableRemoteCommands=1 (如图 7-43 所示)。注意, 这个 Key 如果没有远程执行命令的需要, 为了安全考虑, 可以将其关闭, 在关闭后, Action 中的远程命令也将无法使用, 如果将其开启, 请务必保证 Zabbix-Server 和 Zabbix-Agent 之间的通信安全, 防止被黑客劫持而造成损失。

```
## Option: EnableRemoteCommands
# whether remote commands from zabbix server are allowed.
# 0 - not allowed
# 1 - allowed
# Mandatory: no
# Default:
EnableRemoteCommands=0
```

图 7-43

例如, 执行以下命令:

```
shell# zabbix_get -s 127.0.0.1 -k system.run["ls /"]
bin
boot
dev
etc
home
lib
```

2. Remote command

有关在 Action 中配置远程命令的执行, 请参考 6.3.6 节。

第 8 章 分布式监控

Zabbix 的架构模式分为 Proxy 和 Node，然而，在 Zabbix 2.4 中，已经不再支持 Node 模式了，故本章也不再介绍这种模式。除了架构，被监控端的工作模式（主动模式和被动模式）也是构建分布式监控必须考虑的因素。本章对分布式监控方式进行了详细讲解，构建大型监控环境的读者需深入理解。

8.1 代理架构

Zabbix 是一个分布式的监控系统，这意味着一个中心点、多个分节点的模式可以正常运行。这种情况适合于跨机房、跨地域的网络监控系统。从多个节点收集数据，而每个节点下可以采集多个设备的数据，从而轻松地构建分布式监控系统。

Zabbix 代理（Proxy）可以用在以下环境中。

- 监控远程区域。
- 监控拥有不可靠网络连接的区域。
- 当监控数以千计的设备时分担 Zabbix-Server 服务器的负载。
- 简化分布式监控的维护。

Proxy 架构的使用环境可以用图 8-1 来表示。

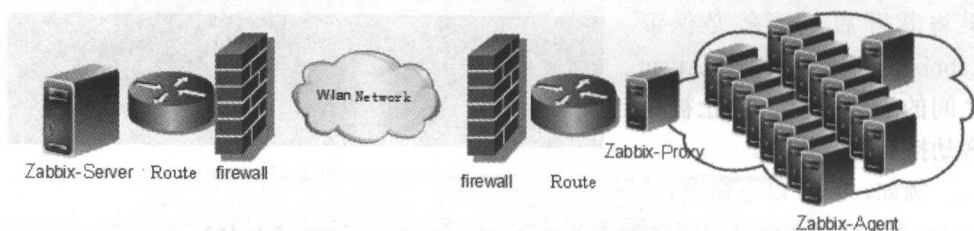


图 8-1

Proxy 和 Server 之间只需要一个 TCP 端口连接。因此，从安全方面考虑，需要一条防火墙规则来确保数据传输的安全。

所有的代理（Proxies）采集到的数据在传送给服务器之前都保存在本地。这样，临时与服务器断开连接也不会导致数据丢失。Proxy 配置文件中的参数 ProxyLocalBuffer 和 ProxyOfflineBuffer 控制数据在本地保存的时间。

1. 安装 Zabbix-Proxy

在用源码安装 Zabbix 的时候（安装的位置与 Zabbix-Server 不在同一台机器中），加上--enable-proxy 参数。

RPM 安装方式下，默认已经开启了此参数，RPM 用 YUM 安装方式的语句如下：

```
shell# rpm -ivh http://repo.zabbix.com/zabbix/2.2/rhel/6/x86_64
/zabbix-release-2.2-1.el6.noarch.rpm
shell# yum install -y zabbix-proxy zabbix-proxy-mysql mysql-server
```

注意：若 zabbix-release 软件的版本有更新，请读者下载并安装最新的版本，地址如下：

```
http://repo.zabbix.com/zabbix/2.2/rhel/6/x86_64/
```

2. 导入 Zabbix-Proxy 的数据库

Zabbix-Proxy 数据库类型可以选择 MySQL、Oracle、SQLite 等，这里采用 MySQL。

Zabbix-Proxy 的数据创建和 Zabbix-Server 相同。

```
shell# service mysqld start
shell# mysql -uroot -p
mysql> use zabbix_proxy;
mysql> create database zabbix_proxy character set utf8;
mysql> grant all privileges on zabbix_proxy.* to zabbix@localhost
identified by 'zabbix';
mysql> flush privileges;
```

与 Zabbix-Server 导入数据唯一的不同的是，前者只需导入 schema.sql 即可。

```
mysql> source /usr/share/doc/zabbix-server-mysql-2.2.2/create/s
chema.sql;
```

官方的 RPM 安装路径如上，如果是其他方式，读者需要输入正确的路径。

3. 配置 zabbix_proxy.conf

zabbix_proxy.conf 重要的参数如表 8-1 所示，读者需要对这些参数进行深入理解。

表 8-1

参 数	参 数 配 置	描 述
ProxyMode	ProxyMode=0	默认参数值为 0，即 Zabbix-Proxy 工作于主动模式
	ProxyMode=1	表示 Zabbix-Proxy 工作于被动模式
Server	Server=X.X.X.X	该参数工作于主动模式（ProxyMode=0），从 X.X.X.X 这个 IP 地址的 Zabbix-Server 获取监控配置信息。被动模式中此参数无效
ServerPort	ServerPort=10051	默认参数为 10051，工作于主动模式，被动模式中此参数无效

续表

参 数	参 数 配 置	描 述
Hostname	Hostname=Zabbix proxy	Zabbix Proxy 的主机名字, 需要注意, 这个名字要具有唯一性, 不能重复, 在配置 Proxy 时, Web 中的配置会用到此参数, 即 Administration-DM-Create proxy-Proxy name, 这个是配置 zabbix-proxy 的重点
HostnameItem	HostnameItem=system.hostname	该参数是在 Hostname 没有定义的时候才会生效
ListenPort	ListenPort=10051	Zabbix-Proxy 的默认端口
SourceIP	SourceIP=X.X.X.X	用于多网卡环境, 指定 Zabbix-Proxy 连接的外网 IP 地址
DBHost	DBHost=localhost	Zabbix-Proxy 的数据库 IP 地址
DBName	DBName=zabbix_proxy	Zabbix-Proxy 的数据库名称
DBUser	DBUser=root	Zabbix-Proxy 的数据库用户名
DBPassword	DBPassword=zabbix	Zabbix-Proxy 的数据库密码
DBSocket	DBSocket=/tmp/mysql.sock	Zabbix-Proxy 的 mysql.sock 文件

4. 启动 Zabbix-Proxy 服务

```
shell# service zabbix-proxy start
shell# chkconfig zabbix-proxy on
```

5. 在 Zabbix GUI 中配置 Zabbix-Proxy

配置步骤为: 单击 Administration→DM, 选择 Proxies→Create proxy, 如图 8-2 所示。

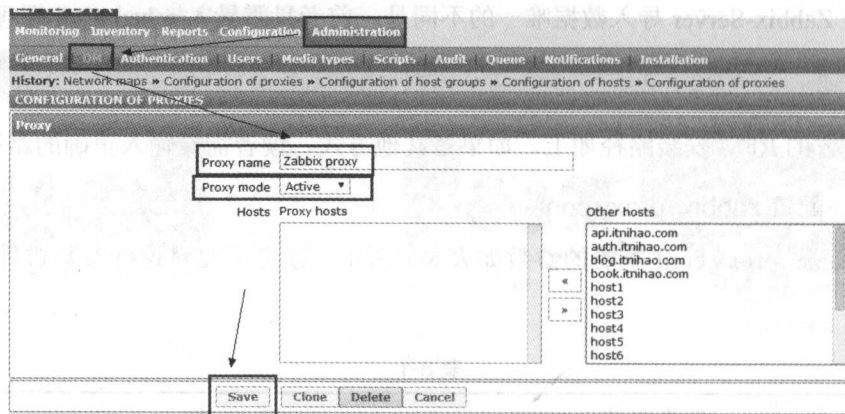


图 8-2

图 8-2 中的各参数说明如下。

- Proxy name, 即Zabbix-Proxy的hostname。
- Proxy mode, 即Zabbix-Proxy的工作模式, 分被动和主动, 默认是主动模式。
- Hosts proxy hosts, 即选择哪些机器处于Proxy模式。

说明: Proxy name 必须和 zabbix_proxy.conf 中的 Hostname 参数一致,即这里的 Proxy name 是来自各个 Node 服务器中定义的 zabbix_proxy.conf 中 Hostname 的参数值,各代理节点的 Hostname 需唯一(假如有 5 个 Proxy,那么配置的时候需要分别添加这 5 个 Proxy,每个 Proxy 的 Hostname 对应 zabbix_proxy.conf 中的 Hostname),这是成功配置 Proxy 的重要参数。

6. 添加 Proxy 架构的主机监控

添加主机的方式和正常模式一样,唯一的不同的是 Monitored by proxy 选项选择的是 Proxy 的名称,如图 8-3 所示。

The screenshot shows the Zabbix web interface for adding a host. The 'Host' tab is selected in the top navigation bar. The 'Host name' field is filled with 'zabbix-agent.itninhao.com'. The 'Visible name' field is empty. Under the 'Groups' section, 'DevOPS' is listed in the 'In groups' box. The 'New group' section is empty. Under 'Agent interfaces', the 'IP address' is '10.10.10.10' and the 'DNS name' is empty. There are 'Add' buttons for 'Agent interfaces', 'SNMP interfaces', 'JMX interfaces', and 'IPMI interfaces'. The 'Monitored by proxy' dropdown is set to 'Zabbix proxy'. The 'Status' dropdown is set to 'Monitored'. At the bottom, there are buttons for 'Save', 'Clone', 'Full clone', 'Delete', and 'Cancel'.

图 8-3

8.2 节点架构

节点架构与 Proxy 相似,唯一的不同的是节点(Node)架构中,Node 本地是一个独立的 Zabbix-Server,可以拥有自己的前端,每个 Node 可以分别进行独立管理,Node 架构适用于这样的环境:一个公司有多个分公司,而每个分公司有自己的 Zabbix Server 管理人员,可以对自己的 Zabbix-Server 进行独立管理,但监控数据需要汇总到总公司。这种架构在实际的生产环境中维护起来比较麻烦,故一般不采用这种架构。

在 Zabbix 2.4 中,由于 Node 模式在大型分布式监控系统架构中存在的问题较

多，将不再支持 Node 架构，仅支持 Proxy 一种分布式架构。
本书将不对这种架构的搭建进行讲解，读者如果有需要，请参考以下官方地址。

https://www.zabbix.com/documentation/2.2/manual/distributed_monitoring/nodes

8.3 被动模式和主动模式

Zabbix-Agent 的工作方式有 Active（主动模式）和 Passive（被动模式）。Zabbix-Server 和 Zabbix-Agent 之间的通信是 Zabbix 的专用协议，数据格式为 JSON。主动模式由于是 Agent 将采集到的数据主动发送给 Server，而不需要 Server 每次连接 Agent 等待采集，所以采用主动模式会使 Zabbix-Server 具有最好的性能。在大型环境中，一定要将工作模式设置为主动模式，并尽可能地采用更多的 Proxy，以降低 Server 的负担。

8.3.1 被动模式

1. 被动模式的配置

默认情况下，Zabbix-Agent 工作在被动模式下，工作的模式是由 Key 和 zabbix_agentd.conf 参数配置决定的。

在 Items 中，将 Items 的检测方式设置为被动模式，如图 8-4 所示（默认模式，无须修改）。

Host

Percona MySQL Server Template

Name

Unflushed Log

Type

Zabbix agent

Key

MySQL.unflushed-log

Information

Numeric (float)

图 8-4

默认是被动模式，如图 8-5 所示。

Key	Interval	History	Trends	Type	Applications
MySQL.unflushed-log	300	90	365	Zabbix agent	MySQL
MySQL.uncheckedpointed-bytes	300	90	365	Zabbix agent	MySQL
proc.num[mysqld]	60	90	365	Zabbix agent	MySQL
MySQL.total-mem-alloc	300	90	365	Zabbix agent	MySQL
MySQL.Threads-running	300	90	365	Zabbix agent	MySQL
MySQL.Threads-created	300	90	365	Zabbix agent	MySQL
MySQL.Threads-connected	300	90	365	Zabbix agent	MySQL
MySQL.Threads-cached	300	90	365	Zabbix agent	MySQL
MySQL.thread-hash-memory	300	90	365	Zabbix agent	MySQL

图 8-5

在 `zabbix_agentd.conf` 中, 设置工作模式为被动模式, 即将 `Server` 参数设置为允许连接的 `Server` (能被数据采集的, 不一定是 `Server`) `IP`。

```
shell# vim /etc/zabbix/zabbix_agentd.conf
### Option: Server
#List of comma delimited IP addresses (or hostnames) of Zabbix servers.
#Incoming connections will be accepted only from the hosts listed here.
#If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally.
#
# Mandatory: no
# Default:
# Server=

Server=127.0.0.1,192.168.0.201
```

2. Server 向 Agent 请求数据

```
<item key>\n
```

3. Agent 向 Server 响应数据

```
<HEADER><DATALEN><DATA>
```

4. 被动模式的流程

被动模式的流程如下。

- ① `Server` 打开一个 `TCP` 连接。
- ② `Server` 发送一个 `key` 为 `agent.ping\n`。
- ③ `Agent` 接收到这个请求, 然后响应数据 `<HEADER><DATALEN>1`。
- ④ `Server` 对接收到的数据进行处理。
- ⑤ `TCP` 连接关闭。

8.3.2 主动模式

1. 主动模式的配置

修改 `zabbix_agentd.conf` 中 `ServerActive=Server` 的 `IP` 地址 (这里的 `Server` 也可以是 `Proxy`、`Node` 的 `IP` 地址), 即可配置主动模式。注意, 修改配置文件后一定要重启 `Zabbix_Agentd` 服务。

```
shell# vim /etc/zabbix/zabbix_agentd.conf
##### Active checks related

### Option: ServerActive
# List of comma delimited IP:port (or hostname:port) pairs of Zabbix servers for active checks.
```



```
# If port is not specified, default port is used.
# IPv6 addresses must be enclosed in square brackets if port
for that host is specified.
# If port is not specified, square brackets for IPv6 addresse
s are optional.
# If this parameter is not specified, active checks are disab
led.
# Example: ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:
30051,::1,[12fc::1]
#
# Mandatory: no
# Default:
# ServerActive=

ServerActive=127.0.0.1:10051,10.10.10.1:10051
```

在 Items 中，将 Items 的检测方式修改为主动模式（默认为被动模式），如图 8-6 所示。如果为了提高性能或者环境需要，将所有的 Items 都设置为主动模式，可以进行批量修改，选择所有的 Items，选择下拉菜单 Mass update，如图 8-7 所示。

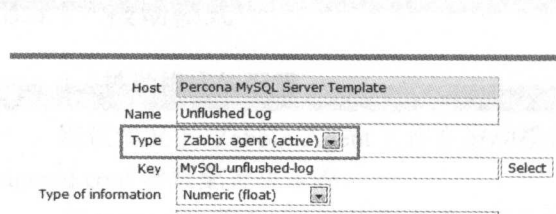


图 8-6

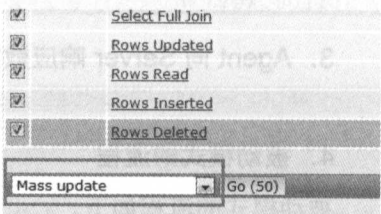


图 8-7

勾选 Type，选择 Zabbix agent (active) 主动模式，如图 8-8 所示。单击“Update”按钮，即可执行批量修改，如图 8-9 所示。

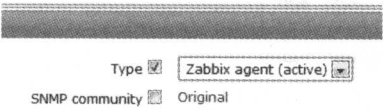


图 8-8

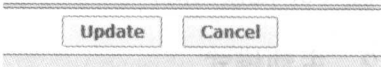


图 8-9

查看 Items，批量修改成功的结果如图 8-10 所示。

Key	Interval	History	Trends	Type
MySQL.unflushed-log	300	90	365	Zabbix agent (active)
MySQL.uncheckpointed-bytes	300	90	365	Zabbix agent (active)
proc.num[mysqld]	60	90	365	Zabbix agent (active)
MySQL.total-mem-alloc	300	90	365	Zabbix agent (active)
MySQL.Threads-running	300	90	365	Zabbix agent (active)
MySQL.Threads-created	300	90	365	Zabbix agent (active)
MySQL.Threads-connected	300	90	365	Zabbix agent (active)
MySQL.Threads-cached	300	90	365	Zabbix agent (active)

图 8-10

2. Agent 向 Server 请求检测列表

(1) Agent 主动向 Server 发送请求

```
<HEADER><DATALEN>{
  "request":"active checks",
  "host":"<hostname>"
}
```

(2) Server 进行响应

```
"response":"success",
"data":[
  {
    "key":"log[\\home\\zabbix\\logs\\zabbix_agentd.log]",
    "delay":"30",
    "lastlogsize":"0"
  },
  {
    "key":"agent.version",
    "delay":"600"
  }
]
```

Zabbix-Server 端必须响应成功，才能返回响应数据。对于返回的每个 Item，Key 和 Delay（间隔的检测时间）必须是存在的，如果 Items 的类型是 Log，则 lastlogsize 参数是必须有的。

如果发送的数据失败（例如主机，Items 被关闭或删除），Agent 不会再重新发送这些数据。

(3) 运行流程

以上运行流程可以表示为如下步骤。

- ① Agent 向 Server 建立一个 TCP 的连接。
- ② Agent 请求需要检测的数据列表。
- ③ Server 响应 Agent，发送一个 Items 列表（item key、delay）。
- ④ Agent 响应请求。
- ⑤ TCP 连接完成本次会话后关闭。
- ⑥ Agent 开始周期性地收集数据。

3. Agent 发送数据给 Server

(1) Agent 端发送数据采集周期内的数据给 Server 端

```
<HEADER><DATALEN>{
  "request":"agent data",
  "data":[
    {
      "host":"<hostname>",
```

```
        "key": "log[\\home\\zabbix\\logs\\zabbix_agentd.log]",
        "value": " 13039:20090907:184546.759 zabbix_agentd starte
d. ZABBIX 1.6.6 (revision {7836}).",
        "lastlogsize": 80,
        "clock": 1252926015
    },
    {
        "host": "<hostname>",
        "key": "agent.version",
        "value": "1.6.6",
        "clock": 1252926015
    }
],
"clock": 1252926016
}
```

(2) Server 端响应

```
<HEADER><DATALEN>{
  "response": "success",
  "info": "Processed 2 Failed 0 Total 2 Seconds spent 0.002070"
}
```

(3) 运行流程

运行流程如下。

- ① Agent 向 Server 建立一个 TCP 连接。
- ② Agent 发送在采集周期内，需要采集数据给 Server。
- ③ Server 处理 Agent 发送的数据。
- ④ TCP 连接关闭。

第 9 章 Zabbix 与自动化运维

在 Zabbix 的自动化中，其网络自动发现、主动注册、LLD 等自带的自动化功能对解决监控自动化运维具有其他软件不可比拟的优势，本章重点介绍这部分内容。同时，考虑到配置文件的管理，本章对 SaltStack 自动化配置工具也进行了讲解，从工具层面展示了自动化运维底层的配置管理。

9.1 监控自动化

监控在运维工作中所占的比例为 30% 左右，监控做得好，会省去很多事情，让工作能有序地进行。理想的监控应该是自动化的，即只需配置规则，即可完成所有的工作，比如主机的自动添加和注册、模板的自动添加、分组的自动添加、出现故障后能自动处理和自动修复。Zabbix 就是具有以上自动化功能的一款监控软件，它如何实现监控自动化功能呢？

首先，Zabbix 提供了网络自动发现功能，该功能可以基于 FTP、SSH、Web、LDAP、POP3、IMAP、SMTP、TCP、SNMP、Telnet、zabbix_agent 等，主动扫描网络中的这些协议和服务，当这些协议和服务存在的时候，即认为主机和设备存在，表示该 IP 存活，而是否添加到监控，是由 Actions 来决定的。在 Zabbix 中，网络发现和自动注册都具有以上提到的功能。

其次，Zabbix 提供了对多变的监控项目自动发现监控的功能，例如，Zabbix 自带有两个网卡，再增加两个网卡，新增加的两个网卡如何做到自动监控？再如，磁盘分区、硬盘设备等，这些量存在不确定的因素，一台服务器可能只有一个硬盘，也可能有多个，如何去自动监控？如上问题，都可以用 Zabbix 的 Low level discovery 功能轻松完成。即对于监控项中具有相同的属性，但存在部分变量配置不同的监控项，完成自动添加监控项。

基于 Zabbix 的这两个功能，我们可以实现以下几点：

- 自动添加主机、自动添加模板、自动分组、自动添加监控项、触发器等。
- 自动添加监控项中有规律的“变量”。

如上所讨论的，我们可以做到对监控的自动化配置，例如：某公司有 20 个机房，每个机房有 300 台服务器，每台服务器运行一个应用，但是每台服务器的应用中，有些端口不同，应该怎么实现监控呢？下面将围绕这个问题进行介绍。

9.2 网络发现

Zabbix 的网络自动发现是一个非常强大的功能，该功能可以完成以下工作。

- 快速发现并添加主机。
- 简单的管理。
- 随着环境的改变而快速搭建监控系统。

网络发现基于以下信息。

- IP地址段。
- 基于服务的FTP、SSH、Web、POP3、IMAP、TCP等。
- 从Zabbix-Agent接收到的信息。
- 从SNMP agent接收到的信息。

网络自动发现功能不能做到的事情是网络拓扑图的发现。

网络自动发现的两个工作流程是：Discovery 和 Actions。

下面以一个例子来介绍如何配置网络发现。

进入 Web 前端，单击 Configuration→Discovery→Create discovery rule，如图 9-1 所示。

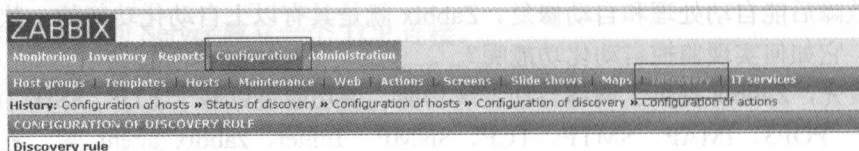


图 9-1

单击 Create discovery rule，输入信息如图 9-2 所示。

 The screenshot shows the 'Create discovery rule' form. The form has several fields: 'Name' (text input, value: server 10.10.10.1), 'Discovery by proxy' (dropdown menu, value: No proxy), 'IP range' (text input, value: 10.10.10.1-254), 'Delay (in sec)' (text input, value: 180), 'Checks' (list of checks: ICMP ping, Zabbix agent "system.uname", with 'Remove' buttons next to each), 'Device uniqueness criteria' (radio buttons: IP address (selected), Zabbix agent "system.uname"), 'Enabled' (checkbox, checked), and a 'Save' button (circled) along with 'Clone', 'Delete', and 'Cancel' buttons.

图 9-2

图 9-2 中, 各参数的说明如下。

- Name: 名称, 可以写自己能代表服务功能的名称, 便于识别。
- Discovery by proxy: 是否通过代理。
- IP range: IP地址的范围, 可以写一段地址, 也可以写多段地址。
- Delay: 检测时间周期, 默认值是3600, 即一小时才能发现服务。
- Checks: 检测命令, 这里选择ICMP, 用Zabbix-Agent来检测。
- Device uniqueness criteria: 设备唯一的名称, 此处采用IP。
- Enabled: 发现功能是否激活。

创建自动添加到相应模板的规则, 如图 9-3 所示。

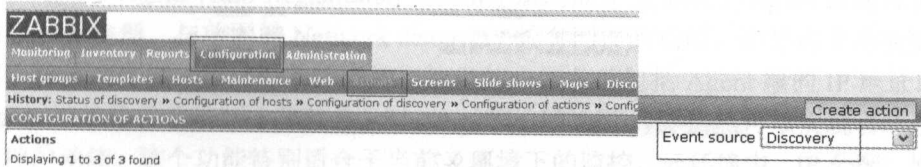


图 9-3

选择 Discovery, 弹出如图 9-4 所示的界面。

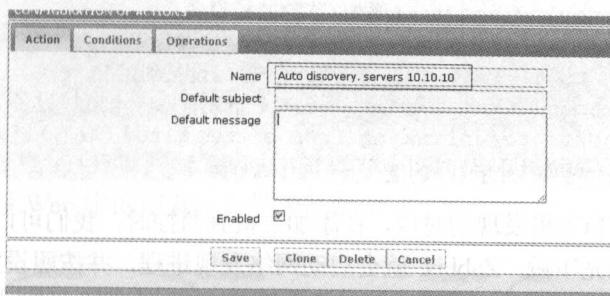


图 9-4

Name 项填写业务相关的 Action 名称, 如图 9-5 所示。

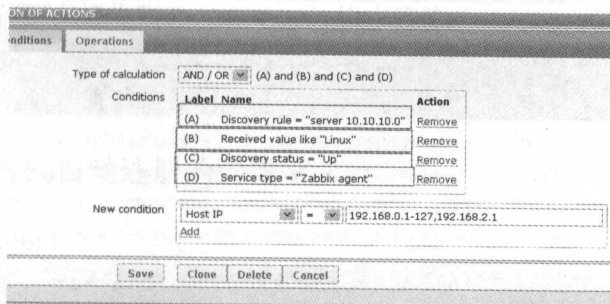


图 9-5

注意: Discovery rule 是添加前面定义的 Discovery rule 项目, 如图 9-6 所示。

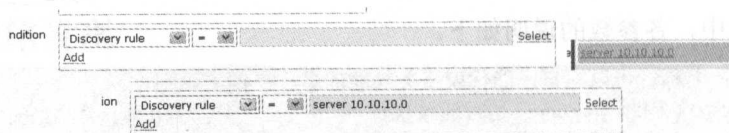


图 9-6

Discovery rule 是添加从自动发现规则中发现的主机，包含如图 9-5 中所配置的这些规则，会进行下一步操作设置，如图 9-7 所示。

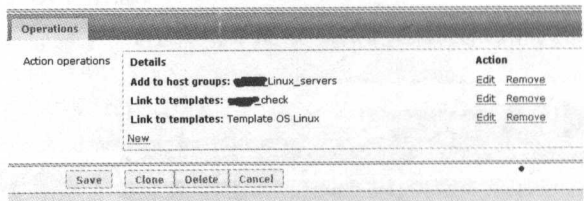


图 9-7

操作中包括以下功能：

- Sending notifications #发送通知
- Adding/removing hosts #增加/移除监控设备
- Enabling/disabling hosts #开启/关闭监控设备
- Adding hosts to a group #增加一个监控设备到组
- Removing hosts from a group #从一个组中移除一个监控设备
- Linking hosts to/unlinking from a template#(取消)链接设备到一个模板
- Executing remote scripts #执行远程脚本

也就是说，当主机发现的时候，在添加主机到监控时，我们可以执行以上操作。当规则添加完毕后，Zabbix 就会启动网络发现进程，并按照设定的扫描规则去发现设备，如图 9-8 所示，是网络发现后的主机（在 **Monitoring→Discovery** 中查看）。

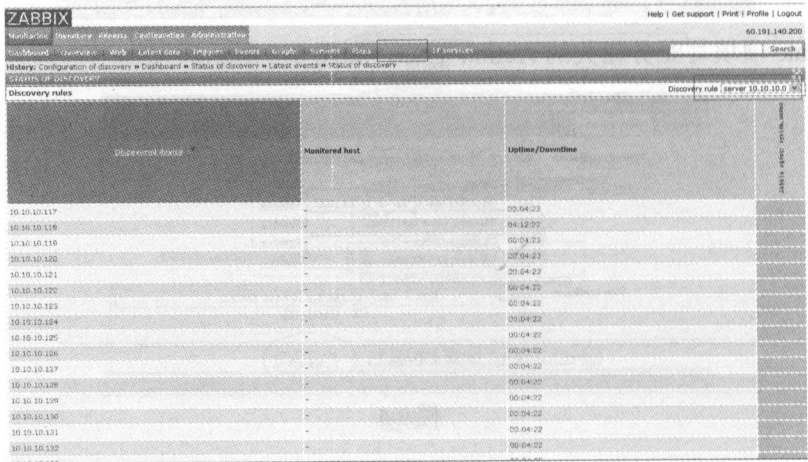


图 9-8

可以看到, Zabbix 的网络自动发现功能的确很强大。在很短的时间内就可以自动添加成百上千台机器的监控, 其自动添加模板和自动分组的功能更是自动化运维中的利器。

9.3 主动方式的自动注册

9.3.1 功能概述

Active Agent Auto-Registration (主动注册功能) 主要用于 Agent 主动且自动向 Server 注册。与前面的 Network discovery 具有同样的功能, 但是这个功能更适用于特定的环境, 当一个条件未知 (这里的未知条件包括 Agent 端的 IP 地址段、Agent 端的操作系统版本等信息) 时, Agent 去请求 Server 仍然可以实现自动添加监控的功能。这个功能特别适合于当前云环境下的监控, 云环境中, IP 分配、操作系统版本等都可能随机, 该功能可以很好地解决类似的问题。

当我们配置一个自动注册功能的时候, 需要对/etc/zabbix/zabbix_agentd.conf 做以下设置:

```
shell# vim/etc/zabbix/zabbix_agentd.conf
ListenIP      #此处填写Agent监听的IP地址
ListenPort    #此处填写Agent监听的端口
```

当添加一个新的自动注册机器时, IP 地址和端口取的是接收到的数据中的信息, 以此来填写 Web 中的配置, 如图 9-9 所示。

图 9-9

Zabbix-Server 在接收到 Zabbix-Agent 的注册请求时, 如果没有收到 IP 地址的值, 则使用 Zabbix-Agent 和 Zabbix-Server 建立 TCP 连接时使用的 IP 地址。如果没有收到 Port 的值, 则默认使用 10050 作为端口。

9.3.2 主动方式自动注册的配置

1. 配置 Agent

```
shell# vim /etc/zabbix/zabbix_agentd.conf
ServerActive=10.0.0.1
Hostname=web1.itnihao.com
```

在 Agent 端配置主动模式, 即配置 ServerActive 参数。注意, 如果没有设置

Hostname, 将会使用默认的主机名, 在 Linux 中使用 Hostname 命令获取到的主机名 (即使用 HostnameItem=system.hostname 这个 Key 来发现 Hostname)。

2. Web 前端配置自动注册功能

下面的例子是在代理的方式下实现自动注册功能, 主要步骤如下。

① 在 Web 前端单击菜单项 **Configuration**→**Actions**, 在 event source 中选择 Auto registration, 单击 Create action。

② 在 Action 选项中配置名称等信息。

③ 选择 Conditions 选项, 在 New condition 下拉框中选择 Proxy, 选择代理的主机, 单击 Add 按钮。

④ 选择 Operations 选项, 选择相关的条件操作, 如 'Add host'、'Add to host groups'、'Link to templates' 等。

3. 配置过程

下面详细介绍如何配置 Proxy 的自动注册功能。

依次选择菜单栏 **Configuration** → **Actions** → **Create action** → **Auto registration**, 如图 9-10 所示。

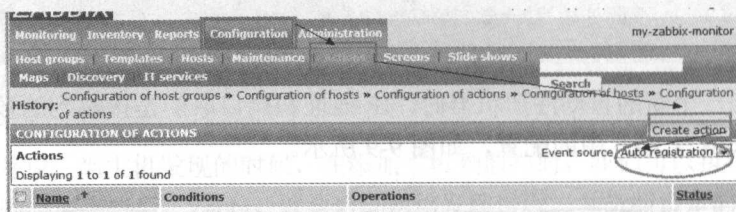


图 9-10

在 Action 选项中输入 Name 的名称, 如图 9-11 所示。

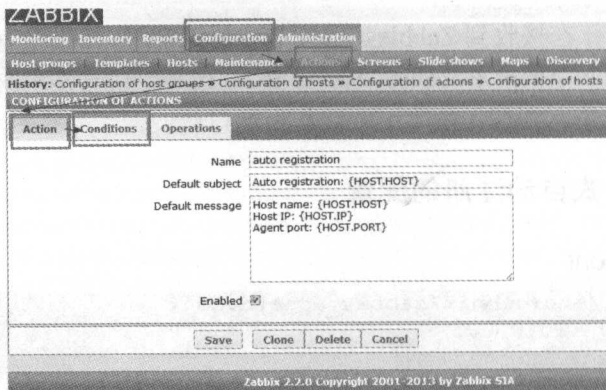


图 9-11

在 Conditions 中选择条件为 Proxy，如图 9-12 所示。

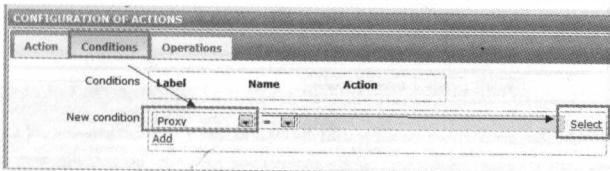


图 9-12

选择我们之前创建的代理，如图 9-13 所示。

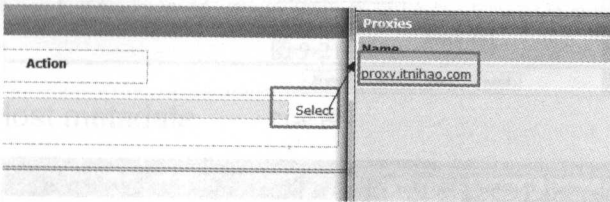


图 9-13

单击 “Add” 按钮添加，如图 9-14 所示。

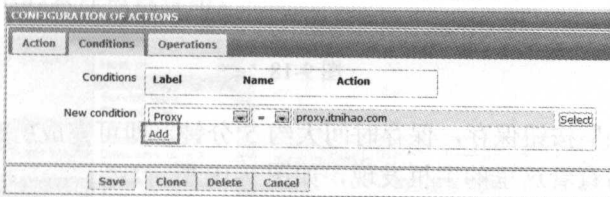


图 9-14

添加好的界面如图 9-15 所示。

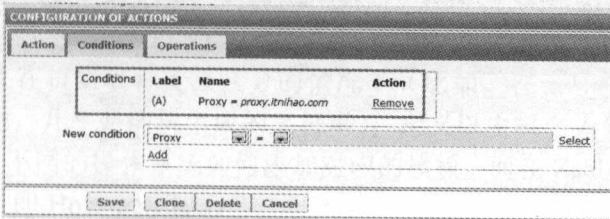


图 9-15

动作、条件都设置好后，下面就是开始操作了，如图 9-16 所示。

在操作中选择添加主机、主机组和模板等，如图 9-17 所示。

添加好的内容如图 9-18 所示（读者会看到这里的模板不是标准名称，而是 Template OS Linux-active，由于自带的模板是被动方式，所以这里修改了模板，设置为主动模式。关于主动模式，请参考第 8 章）。

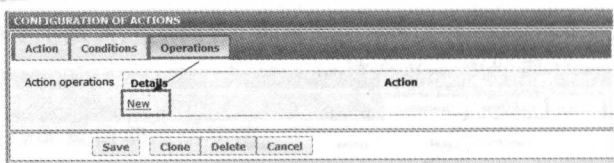


图 9-16

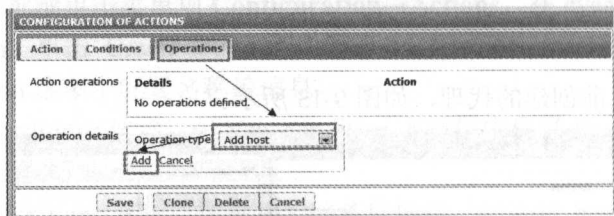


图 9-17

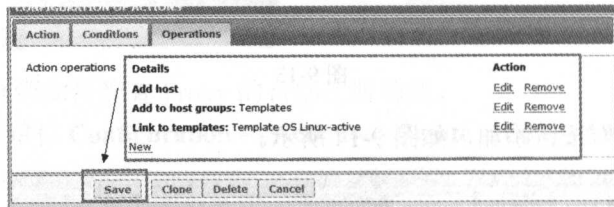


图 9-18

单击“Save”按钮保存，保存时间大约 5 分钟，即可完成所有网段主机的自动发现。完成后查看这里的主机发现，如图 9-19 所示。

ZABBIX

Help | Get support | Print | Profile | Logout

MonitoringInventoryReportsConfigurationAdministration

my-zabbix-monitor

DashboardOverviewWebLatest dataTriggersEventsGraphsScreensMapsDiscoveryIT services

Search

History: Configuration of actions » Configuration of hosts » Configuration of actions » Configuration of hosts » Dashboard

STATUS OF DISCOVERY

Discovery rule: proxy.itnihao.com

Discovery rules

Discovered device	Monitored host	Uptime/Downtime	ICMP ping
proxy.itnihao.com (40 devices)			
192.168.151.1	-	01:11:39	
192.168.151.5	-	02:10:57	
192.168.151.11	-	02:10:26	
192.168.151.12	-	00:10:44	
192.168.151.15	-	00:10:29	
192.168.151.17	-	00:10:18	
192.168.151.192	-	01:58:51	
192.168.151.173	-	00:55:21	
192.168.151.180	-	00:54:42	
192.168.151.181	-	01:54:23	
192.168.151.183	-	01:54:13	
192.168.151.185	-	01:54:02	
192.168.151.192	-	00:53:44	

图 9-19

符合条件的主机会自动添加到监控项中，devops.itnihao.com 主机已经被监控到，如图 9-20 所示，从图 9-21 中可看到已经添加监控完毕。

192.168.151.200 (devops.itnihao.com) devops.itnihao.com 02:10:24

图 9-20

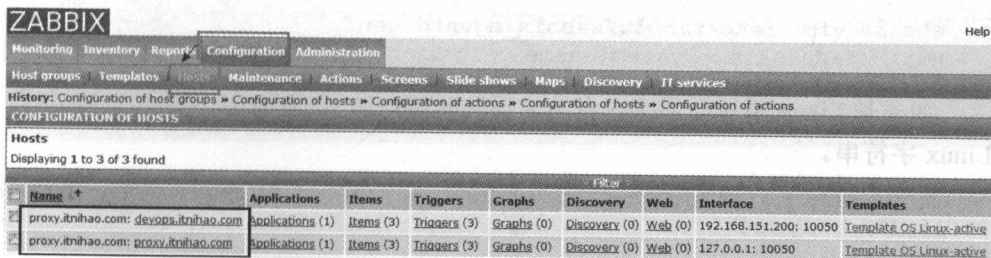


图 9-21

9.3.3 使用 Host metadata

Host metadata 是 Zabbix 2.2 新增加的功能，该功能在 Zabbix-Agent 端可以自定义条件，在选择自动注册的时候，Zabbix-Server 端可以根据 Host metadata 来选择条件，从而实现更多的条件筛选。在 Zabbix 2.2 之前的版本中，在网络发现中有如图 9-22 所示的条件可以使用。

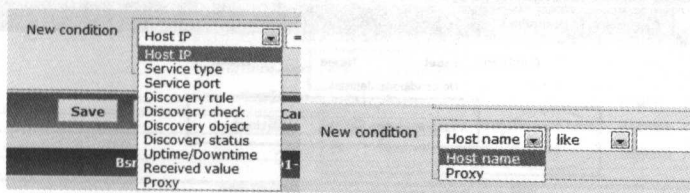


图 9-22

在 Zabbix 2.2 以前的版本中，在自动注册选项中只有两个条件可供选择，一个是 Host name，另一个是 Proxy，如图 9-22 所示，在实际的生产环境中并不能满足需求，如果在机器名称无法区分的情况下（比如，在云环境中，主机名通常没有规律可循，几乎是随机生成的字符串），定义以主机名为条件，会出现很多问题，例如：不同的操作系统如何添加对应的模板。而在 Zabbix 2.2 中给出了新的解决方案，即 Host metadata。

在 zabbix_agentd.conf 中，新增了以下两个配置参数。

- HostMetadata=字符串：长度范围为 0~255。
- HostMetadataItem=item：用于 Item 获取数据。这里的 Item 可以设置为 system.uname，或者是其他获取到的字符串，长度限制为 0~255。

下面分别对这两个参数（HostMetadata 和 HostMetadataItem）进行详解。

1. 使用 HostMetadataItem 的配置参数

在 Zabbix-Agent 端配置, 语句如下:

```
shell# vim /etc/zabbix/zabbix_agentd.conf
HostMetadataItem=system.uname
```

如果是 Linux 系统, 那么在自动注册的时候, Host metadata 数据中将会包含 Linux 字符串。

```
Linux proxy.itnihao.com 2.6.32-358.el6.x86_64 #1 SMP Fri Feb 22 00:31:26 UTC 2013 x86_64
```

如果是 Windows 系统, 那么在自动注册的时候, Host metadata 数据中将会包含 Windows 字符串。

```
Windows pc 6.1.7601 Microsoft Windows 7 Ultimate Edition Service Pack 1 x64
```

前面我们讨论了 Zabbix-Agent 端的配置, 下面在 Zabbix-Server 的 Web 前端进行配置。

1) 对 Linux 系统的规则添加, 如图 9-23 所示。

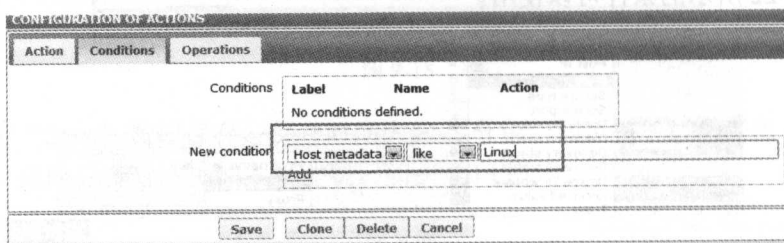


图 9-23

名字: Linux host autoregistration

条件: Host metadata like Linux

操作: Link to templates: Template OS Linux

2) 对 Windows 系统的模板添加。

如果 HostMetadataItem=system.uname 检测出来的值 Windows 的机器, 则在前端 Web 的配置如下:

名字: Windows host autoregistration

条件: Host metadata like Windows

操作: Link to templates: Template OS Windows

2. 使用 HostMetadata 的配置参数

如果在 Zabbix-Agent 端选择使用 HostMetadata 这个配置参数, 在 Zabbix-Server 的 Web 前端配置方法与上面的步骤 1) 和 2) 类似, 则在 Zabbix-

Agent 端配置如下:

```
shell# vim /etc/zabbix/zabbix_agentd.conf
HostMetadata=Linux linux_host
```

在 Windows 下的客户端修改为:

```
/etc/zabbix/zabbix_agentd.conf
HostMetadata=Windows Windows host
```

在 Web 前端设置规则为: 在选择条件的时候, 可以选择两个 (Linux 和 linux_host、Windows 和 Windows host) 同时成立, 如图 9-24 和图 9-25 所示。

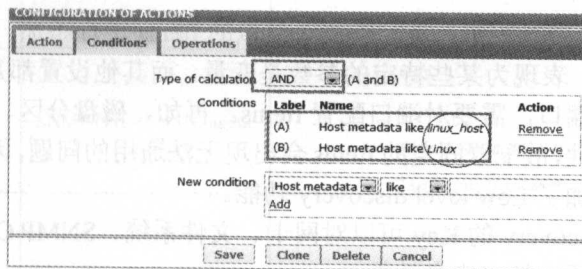


图 9-24

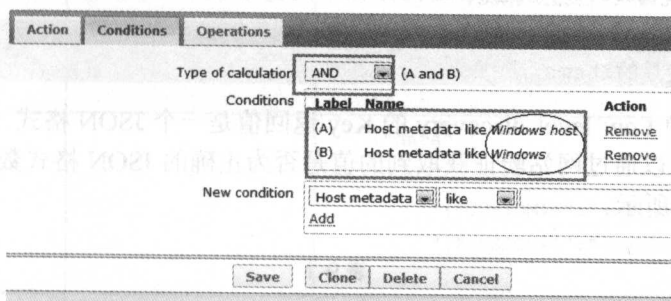


图 9-25

从上面的例子可以看到, 基于 Host metadata 在自动注册中, 我们同样可以有效地设置机器的分组、机器模板的添加等功能。

9.3.4 关于自动注册的注意事项

从前面的内容可知, 自动发现的功能非常强大, 能进一步减轻监控的工作量。若大批量的机器需要添加监控项, 利用自动发现功能就可以实现。

zabbix_proxy.conf 默认只是最基本的配置, 因此要调整为生产环境适用的参数。

对于 zabbix_proxy.conf, 请关注以下参数。

```
ProxyLocalBuffer
ProxyOfflineBuffer
HeartbeatFrequency
```

```
ConfigFrequency
StartPollers
StartIPMIPollers
StartPollersUnreachable
StartTrappers
StartPingers
StartDiscoverers
```

关于这部分参数的调整原则，请读者参考第 12 章。

9.4 Low level discovery 功能

在配置 Items 的过程中，有时需要对类似的 Items 进行添加，这些 Items 具有一些共同的特性，表现为某些特定的参数是变量，而其他设置都是一样的。例如，一个程序有多个端口，需要对端口配置 Items。再如，磁盘分区、网卡的名称等，由于具有不确定性，故配置固定的 Items 会出现无法通用的问题，因而在 Zabbix 2.0 以后的版本中增加了 Low level discovery 功能。

Low level discovery 的 Key 可以对网卡、文件系统、SNMP OIDS 进行自动发现。除此之外，还支持自定义功能。

Low level discovery 的使用过程分以下两步。

- 自动发现特定变量的名称。
- 添加对变量的 Items。

Zabbix 中 Low level discovery 的 Key 返回值是一个 JSON 格式（如果是自定义的规则，可以通过网站验证获取到的值是否为正确的 JSON 格式数据），自带的 Key 如表 9-1 所示。

表 9-1

Key 名称	适用监控方式类型	返回数据格式
vfs.fs.discovery	Zabbix Agent	<pre>{ "data": [{ "{#FSNAME}": "<path>", "{#FSTYPE}": "<fstype>" }, { "{#FSNAME}": "<path>", "{#FSTYPE}": "<fstype>" }, ...] }</pre>

续表

Key 名称	适用监控方式类型	返回数据格式
net.if.discovery	Zabbix Agent	<pre>{ "data": [{ "{#IFNAME}": "<name>" }, { "{#IFNAME}": "<name>" }, ...]}</pre>
snmp.discovery	SNMP (v1, v2 or v3) agent	<pre>{ "data": [{ "{#SNMPINDEX}": "<idx>", "{#SNMPVALUE}": "<value>" }, { "{#SNMPINDEX}": "<idx>", "{#SNMPVALUE}": "<value>" }, ...]}</pre>
custom.discovery	any	<pre>{ "data": [{ "{#CUSTOM1}": "<value>", "{#CUSTOM2}": "<value>" }, { "{#CUSTOM1}": "<value>", "{#CUSTOM2}": "<value>" }, ...]}</pre>

用 Zabbix-Get 来查看 Key 获取到的数据，如图 9-26 所示。

对于 SNMP，不能通过 Zabbix-Get 来验证，只能在 Web 页面中进行配置使用，如图 9-27 所示。

```
[root@zabbix ~]# zabbix_get -s 127.0.0.1 -k net.if.discovery
{
  "data": [
    {
      "#IFNAME": "lo",
    },
    {
      "#IFNAME": "eth2",
    },
    {
      "#IFNAME": "eth0",
    },
    {
      "#IFNAME": "eth1"
    }
  ]
}

[root@zabbix ~]# zabbix_get -s 127.0.0.1 -k vfs.fs.discovery
{
  "data": [
    {
      "#FSNAME": "\\",
      "#FSTYPE": "rootfs",
    },
    {
      "#FSNAME": "\/proc",
      "#FSTYPE": "proc",
    },
    {
      "#FSNAME": "\/sys",
      "#FSTYPE": "sysfs",
    },
    {
      "#FSNAME": "\/dev",
      "#FSTYPE": "devtmpfs",
    },
    {
      "#FSNAME": "\/dev/pts",
      "#FSTYPE": "devpts",
    },
    {
      "#FSNAME": "\/dev/shm",
      "#FSTYPE": "tmpfs",
    },
    {
      "#FSNAME": "\",
      "#FSTYPE": "ext4",
    },
    {
      "#FSNAME": "\/proc/bus/usb",
      "#FSTYPE": "usbfs",
    },
    {
      "#FSNAME": "\/boot",
      "#FSTYPE": "ext4",
    },
    {
      "#FSNAME": "\/proc/sys/fs/binfmt_misc",
      "#FSTYPE": "binfmt_misc"
    }
  ]
}
```

图 9-26

History: Configuration of discovery rules » Configuration of item prototypes » Configuration of discovery rules » Configuration of item prototypes » Configuration of discovery rules

CONFIGURATION OF ITEM PROTOTYPES

Item prototypes of Network Interfaces

Displaying 1 to 8 of 8 found

• Template list Template: Template SNMP OS Linux • Discovery list Discovery: Network interfaces Item prototypes (8) Triggers prototypes (1) Graph prototypes (1) Host prot

Name	Key	Interval	History	Trends	Type
Template SNMP Interfaces: Admin status of interface {#SNMPVALUE}	#AdminStatus[{#SNMPVALUE}]	60	7	365	SNMPv2 agent
Template SNMP Interfaces: Alias of interface {#SNMPVALUE}	#Alias[{#SNMPVALUE}]	3600	7		SNMPv2 agent
Template SNMP Interfaces: Description of interface {#SNMPVALUE}	#Descr[{#SNMPVALUE}]	3600	7		SNMPv2 agent
Template SNMP Interfaces: Inbound errors on interface {#SNMPVALUE}	#InErrors[{#SNMPVALUE}]	60	7	365	SNMPv2 agent
Template SNMP Interfaces: Incoming traffic on interface {#SNMPVALUE}	#InOctets[{#SNMPVALUE}]	60	7	365	SNMPv2 agent
Template SNMP Interfaces: Operational status of interface {#SNMPVALUE}	#OperStatus[{#SNMPVALUE}]	60	7	365	SNMPv2 agent
Template SNMP Interfaces: Outbound errors on interface {#SNMPVALUE}	#OutErrors[{#SNMPVALUE}]	60	7	365	SNMPv2 agent
Template SNMP Interfaces: Outgoing traffic on interface {#SNMPVALUE}	#OutOctets[{#SNMPVALUE}]	60	7	365	SNMPv2 agent

图 9-27

9.4.1 现实案例需求

下面通过一个例子详细介绍如何配置 Low level discovery。

1. 业务需求

现在有大量的 URL 需要监控，形式如 <http://www.itnihao.com>，要求 URL 状态不为 200 时发出告警。

2. 需求详细分析

具有大量的 URL，且 URL 经常变化，要求增加 URL 即可完成监控。

3. 解决方案

分析需求后，发现 URL 是经常变化的，但其他状态不发生改变，那么它刚好可以用 Zabbix 的 Low level discovery 功能轻松解决此问题。

9.4.2 Zabbix 客户端配置

Zabbix 客户端配置语句如下。

```
shell# egrep -v "(#|^$)" /etc/zabbix/zabbix_agentd.conf
LogFile=/var/log/zabbix/zabbix_agentd.log
EnableRemoteCommands=0
Server=127.0.0.1,192.168.0.240      #Zabbix-Server端的IP地址
StartAgents=8
ServerActive=192.168.0.240:10051  #Zabbix-Server端的IP地址
Hostname=node1
Timeout=30
Include=/etc/zabbix/zabbix_agentd.conf.d/ #子配置文件路径
UnsafeUserParameters=1             #自定义的Key中可以包括特殊字符
```

9.4.3 Low level discovery 自动发现脚本编写

Low level discovery 的脚本是一个 JSON 格式，鉴于大多数用户习惯使用 Shell，故此处采用 Shell 来编写，若用 PERL、Python，则代码会更简洁，读者如有需要，可以到 <https://github.com/itnihao/zabbix-book> 中下载相关脚本。

```
shell#cat /etc/zabbix/scripts/web_site_code_status
#!/bin/bash

# function:monitor web status from zabbix
# License: GPL
# mail:itnihao@qq.com
# version:1.0 date:2012-12-09
source /etc/bashrc >/dev/null 2>&1
source /etc/profile >/dev/null 2>&1

#/usr/bin/curl -o /dev/null -s -w %{http_code} http://$1/
Web_SITE_discovery () {
    Web_SITE=$(cat /etc/zabbix/scripts/Web.txt|grep -v "^#")
    printf '{\n'
    printf '\t"data":[\n'
    for((i=0;i<${#Web_SITE[@]};++i))
    {
        num=$(echo ${${#Web_SITE[@]}-1})
    }
}
```



```

        if [ "$i" != ${num} ];
        then
            printf "\t\t{ \n"
            printf "\t\t\t\"{#SITENAME}\":\"${Web_SITE[$i]}\",\n"
        else
            printf "\t\t{ \n"
            printf "\t\t\t\"{#SITENAME}\":\"${Web_SITE[$num]}\",\n"
        fi
    }
}

web_site_code () {
    /usr/bin/curl -o /dev/null -s -w %{http_code} http://$1
}

case "$1" in
web_site_discovery)
    Web_SITE_discovery
;;
web_site_code)
    web_site_code $2
;;
*)
echo "Usage:$0 {web_site_discovery|web_site_code [URL]}"
;;
esac

```

输出的数据为 JSON 格式，如果不能确定输出是否正确，可以通过 JSON 在线验证网站来验证 JSON 数据格式是否正确。输出格式如下。

```

shell# sh web_site_code_status web_site_discovery
{
    "data":[
        {
            "{#SITENAME}":"www.itnihao.com"},
        {
            "{#SITENAME}":"www.baidu.com"},
        {
            "{#SITENAME}":"www.weibo.com"}
    ]
}

```

如果此处采用 Python 来编写，则代码如下。

```

#!/usr/bin/env python
# coding=utf8
# Last modified: 2013-04-12 14:47
# Author: itnihao
# Mail: itnihao@qq.com

import os
import json

r=open('Web.txt','r').read().split()
devices = []

for devpath in r:
    device = os.path.basename(devpath)

```

```

        devices += [{'#SITENAME}':device]}

print json.dumps({'data':devices},sort_keys=True,indent=7,separators=(',',':'))

```

输出格式如下。

```

{
    "data":[
        {
            "#{SITENAME}":"www.itnihao.com"
        },
        {
            "#{SITENAME}":"www.baidu.com"
        },
        {
            "#{SITENAME}":"www.weibo.com"
        }
    ]
}

```

9.4.4 自定义 Key 配置文件

自定义 Key 配置文件如下。

```

shell# cat /etc/zabbix/zabbix_agentd.conf.d/web_site_discovery.conf
UserParameter=web.site.discovery,/etc/zabbix/scripts/web_site_code_status web_site_discovery
UserParameter=web.site.code[*],/etc/zabbix/scripts/web_site_code_status web_site_code $1

```

域名如下。

```

shell# cat /etc/zabbix/scripts/Web.txt
http://www.itnihao.com
http://www.baidu.com
http://www.weibo.com

```

测试语句如下。

```

shell# zabbix_get -s 127.0.0.1 -k web.site.discovery
{
    "data":[
        {
            "#{SITENAME}":"www.itnihao.com"},
        {
            "#{SITENAME}":"www.baidu.com"},
        {
            "#{SITENAME}":"www.weibo.com"}}
}

```

```

shell# zabbix_get -s 127.0.0.1 -k web.site.code[www.itnihao.com]
200
shell# zabbix_get -s 127.0.0.1 -k web.site.code[www.baidu.com]
200
shell# zabbix_get -s 127.0.0.1 -k web.site.code[www.weibo.com]
200

```

URL 的返回状态都为 200。

至此，脚本、客户端配置文件完成。

对以上修改的文件列个清单，具体如下。

```
/etc/zabbix/zabbix_agentd.conf          #Agent配置文件
/etc/zabbix/scripts/web_site_code_status #权限755
/etc/zabbix/scripts/Web.txt              #网站URL存放文件
/etc/zabbix/zabbix_agentd.conf.d/web_site_discovery.conf#子配置文件
```

9.4.5 Web 页面添加 Low level discovery

由于下面的 Web 页面步骤较多，故读者需要细心参考，确保步骤的正确性。

登录 Zabbix 的 Web 页面，创建模板，如图 9-28 所示，步骤如下。

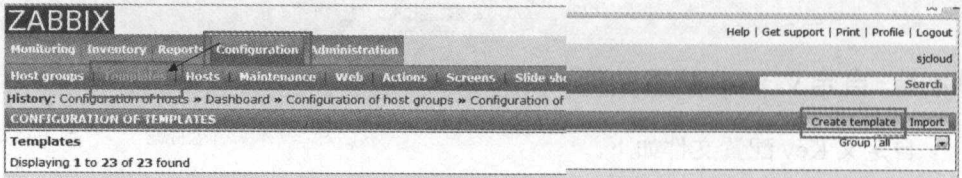


图 9-28

单击 Configuration→Templates→Create templates，出现如图 9-29 所示的配置。

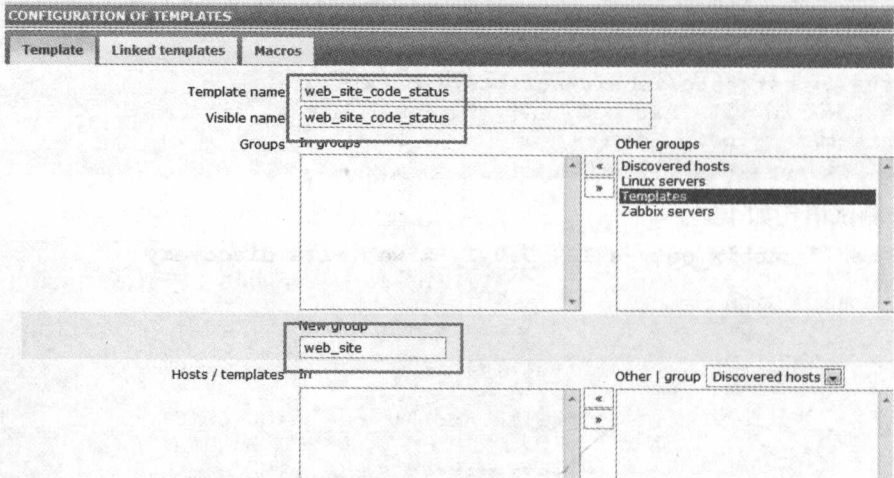


图 9-29

单击 Save 保存，出现如图 9-30 所示的界面。

单击 Application，跳转到如图 9-31 所示的界面，单击 Create application 按钮。

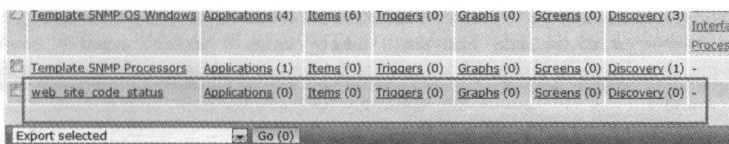


图 9-30

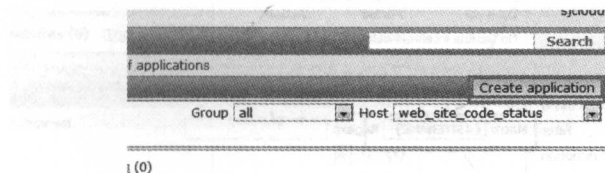


图 9-31

设置组的名称为 `web_site_code_status`，如图 9-32 所示。

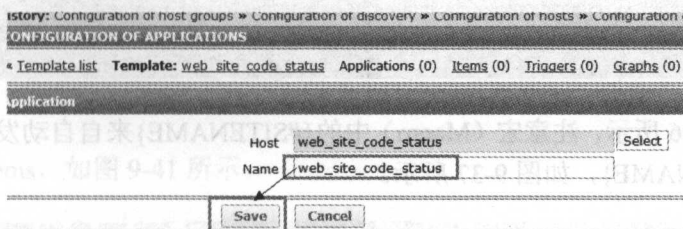


图 9-32

单击 Discovery rules，如图 9-33 所示。

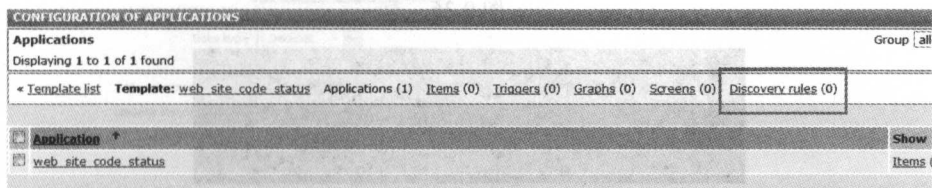


图 9-33

单击 Create discovery rule，选择创建发现规则，如图 9-34 所示。

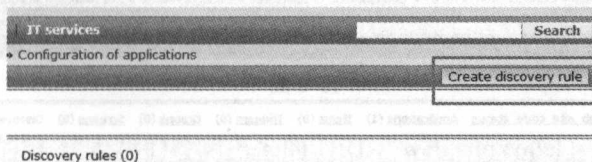


图 9-34

依次填写以下内容，如图 9-35 所示。

CONFIGURATION OF DISCOVERY RULES

◀ Template list Template: web_site_code_status Applications (1) Items (0) Triggers (0) Graphs (0) Screens (0) Discovery rules (0)

Discovery rule

Name: web.site.discovery

Type: Zabbix agent

Key: web.site.discovery

Update interval (in sec): 30

Flexible intervals

Interval	Period	Action
No flexible intervals defined.		

New flexible interval Interval (in sec): 50 Period: 1-7,00:00-24:00 Add

Keep lost resources period (in days): 30

Filter Macro: {#SITENAME} Regexp:

Description:

Status: Enabled

Save Cancel

图 9-35

如图 9-36 所示, 注意宏 (Macro) 中的 {#SITENAME} 来自自动发现脚本中的变量 {#SITENAME}, 如图 9-37 所示。

New flexible interval Interval (in sec): 50 Period: 1-7,00:00-24:00

Keep lost resources period (in days): 30

Filter Macro: {#SITENAME} Regexp:

Description:

图 9-36

```
[root@devops zabbix]# zabbix_get -s 127.0.0.1 -k web.site.discovery
{"data":{
  {
    "{#SITENAME}": "www.itn1hao.com",
  },
  {
    "{#SITENAME}": "www.baidu.com",
  },
  {
    "{#SITENAME}": "www.weibo.com"
  }}}
[root@devops zabbix]#
```

图 9-37

Discovery rules 添加完成, 如图 9-38 所示, 需要进一步添加 Items 等项。

History: Configuration of discovery ▶ Configuration of hosts ▶ Configuration of templates ▶ Configuration of applications ▶ Configuration of discovery rules

Details Discovery rule created

CONFIGURATION OF DISCOVERY RULES

Discovery rules

Displaying 1 to 1 of 1 found

◀ Template list Template: web_site_code_status Applications (1) Items (0) Triggers (0) Graphs (0) Screens (0) Discovery rules (1)

Name	Items	Triggers	Graphs	Key	Interval
web.site.discovery	Item prototypes (0)	Trigger prototypes (0)	Graph prototypes (0)	web.site.discovery	30

Enable selected Go (0)

图 9-38

创建原型 Item，单击 Item prototypes，出现如图 9-39 所示的界面。

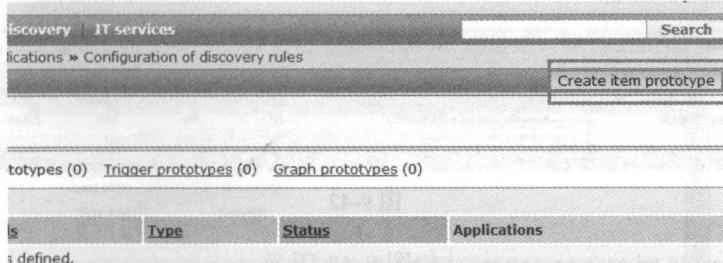


图 9-39

注意，下面开始使用检测 URL 状态的 Key，如图 9-40 所示。

```
[root@devops zabbix]# zabbix_get -s 127.0.0.1 -k web.site.code[www.itnihao.com]
200
[root@devops zabbix]# zabbix_get -s 127.0.0.1 -k web.site.code[www.baidu.com]
200
[root@devops zabbix]# zabbix_get -s 127.0.0.1 -k web.site.code[www.weibo.com]
200
[root@devops zabbix]#
```

图 9-40

添加 Items，如图 9-41 所示。

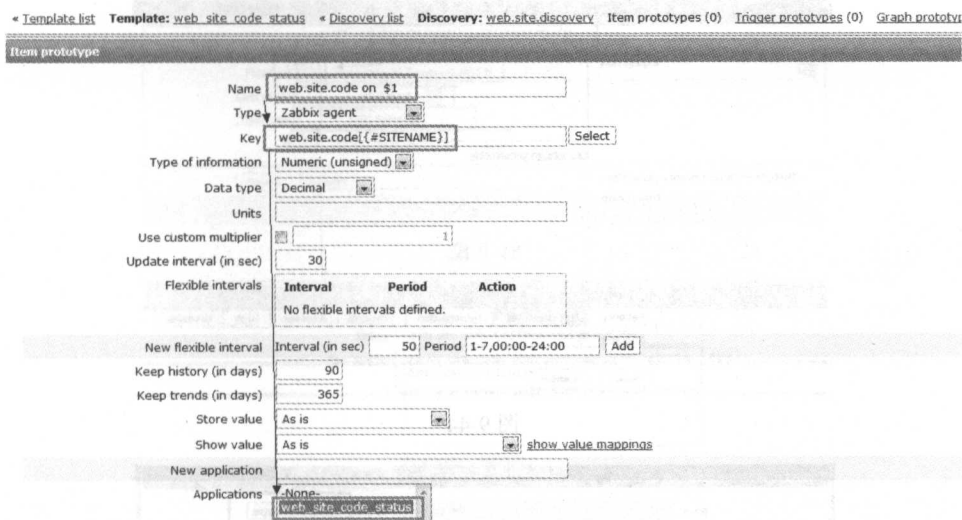


图 9-41

图 9-41 中的 \$1 代表我们检测出来的 URL，web.site.code[#{SITENAME}] 中的变量会依次检测这几个 URL。

单击 Save 保存。创建 Trigger，单击 Trigger prototypes，如图 9-42 所示。

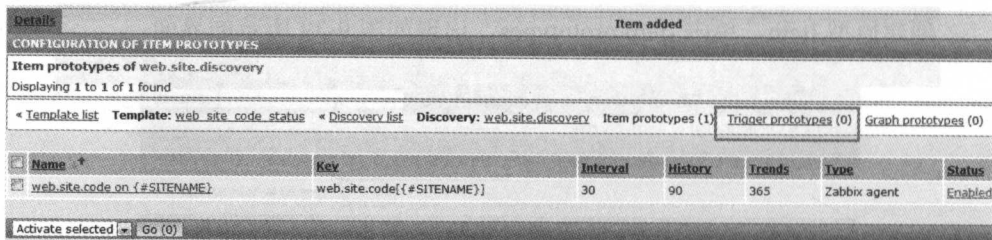


图 9-42

单击 Create trigger prototype, 如图 9-43 所示。

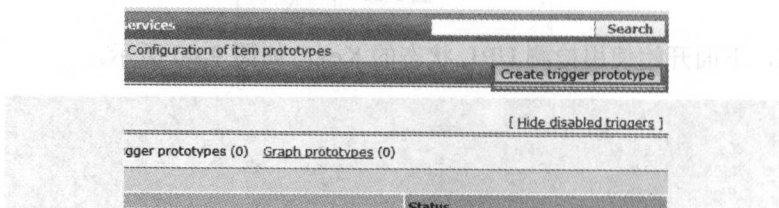


图 9-43

继续配置 Trigger, 如图 9-44 至图 9-49 所示。

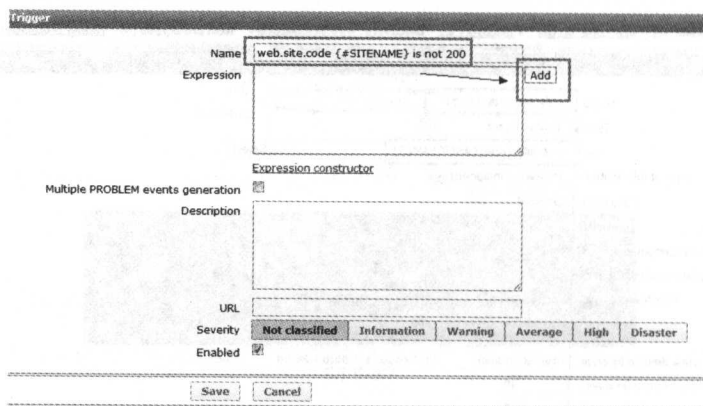


图 9-44

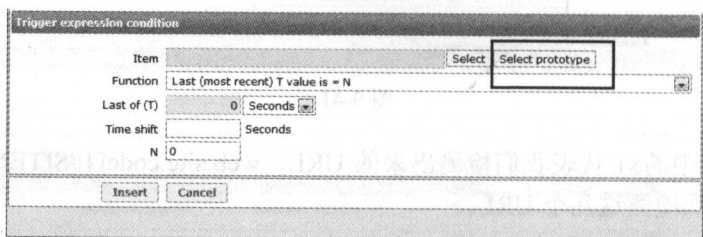


图 9-45

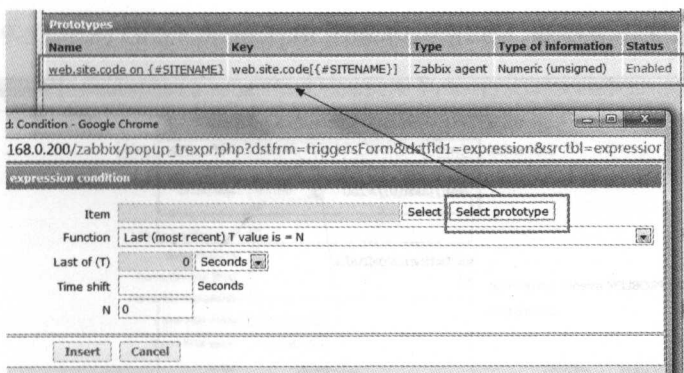


图 9-46

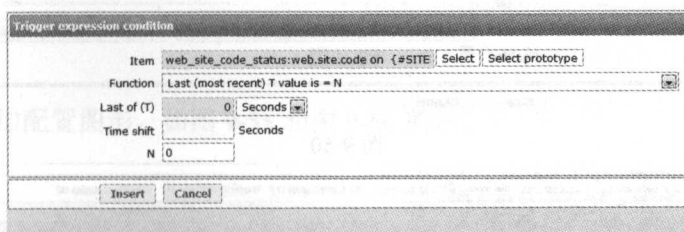


图 9-47

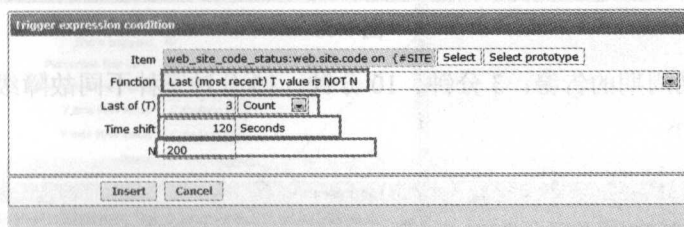


图 9-48

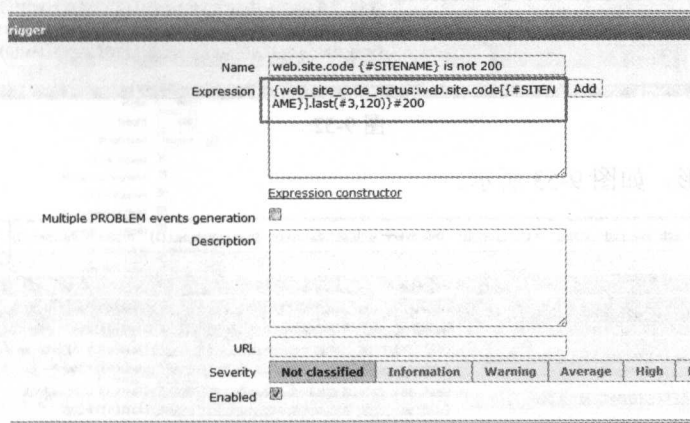


图 9-49

设置多梯度告警检测，如图 9-50 和图 9-51 所示。

图 9-50

Template list Template: web_site_code_status Discovery list Discovery: web.site.discovery Item prototypes (1) Trigger prototypes (1) Graph prototypes (0)				
Severity	Name	Expression	Status	
Information	web.site.code {#SITENAME} is not 200	{web_site_code_status:web.site.code[{#SITENAME}].last({#3,120})}>200&{web_site_code_status:web.site.code[{#SITENAME}].last(0)}>200	Enabled	

图 9-51

设置不同周期的告警：3 分钟、10 分钟、15 分钟三种不同故障级别的告警，如图 9-52 所示。

Severity	Name	Expression
Warning	web.site.code {#SITENAME} in 10 minute is not 200	{web_site_code_status:web.site.code[{#SITENAME}].last({#3,600})}>200&{web_site_code_status:web.site.code[{#SITENAME}].last(0)}>200
Average	web.site.code {#SITENAME} in 15 minute is not 200	{web_site_code_status:web.site.code[{#SITENAME}].last({#3,900})}>200&{web_site_code_status:web.site.code[{#SITENAME}].last(0)}>200
Information	web.site.code {#SITENAME} is not 200	{web_site_code_status:web.site.code[{#SITENAME}].last({#3,120})}>200&{web_site_code_status:web.site.code[{#SITENAME}].last(0)}>200

图 9-52

添加图形，如图 9-53 所示。

Template list Template: web_site_code_status Discovery list Discovery: web.site.discovery Item prototypes (1) Trigger prototypes (3) Graph prototypes (0)				
Severity	Name	Expression		
Warning	web.site.code {#SITENAME} in 10 minute is not 200	{web_site_code_status:web.site.code[{#SITENAME}].last({#3,600})}>200&{web_site_code_status:web.site.code[{#SITENAME}].last(0)}>200		
Average	web.site.code {#SITENAME} in 15 minute is not 200	{web_site_code_status:web.site.code[{#SITENAME}].last({#3,900})}>200&{web_site_code_status:web.site.code[{#SITENAME}].last(0)}>200		
Information	web.site.code {#SITENAME} is not 200	{web_site_code_status:web.site.code[{#SITENAME}].last({#3,120})}>200&{web_site_code_status:web.site.code[{#SITENAME}].last(0)}>200		

图 9-53

单击图 9-53 中的 Graph prototypes，出现如图 9-54 所示的界面。

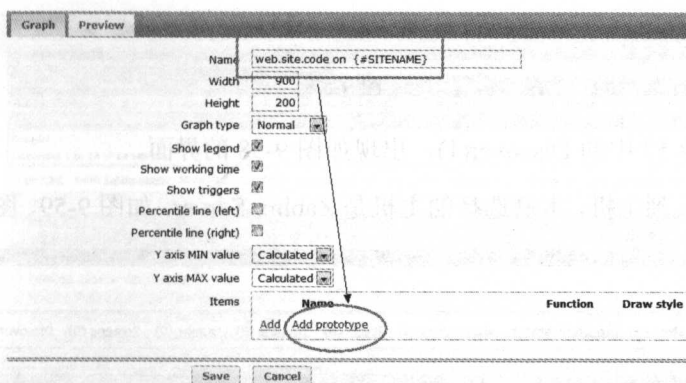


图 9-54

继续添加配置图形，如图 9-55 和图 9-56 所示。

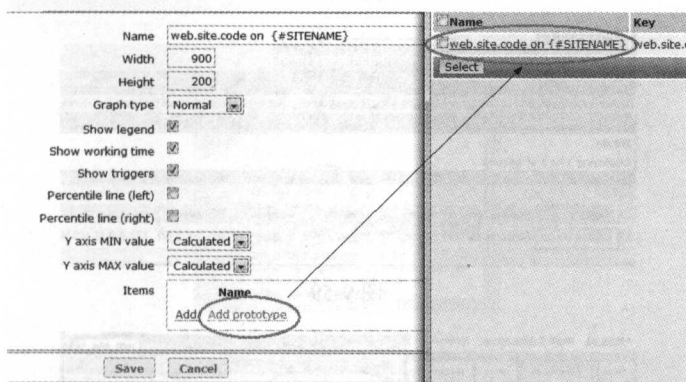


图 9-55

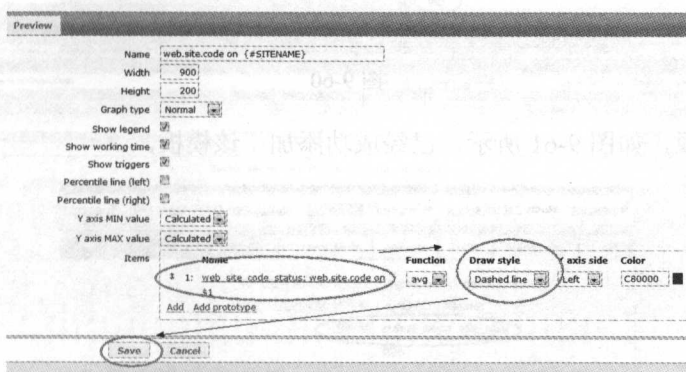


图 9-56

模板创建成功，如图 9-57 所示。

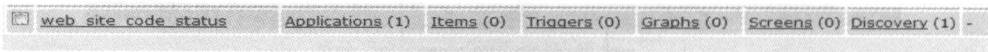


图 9-57

单击图 9-57 中的 Discover(1)，出现如图 9-58 的界面。

添加模板到主机，本例选择的主机是 Zabbix-Server，如图 9-59、图 9-60 所示。

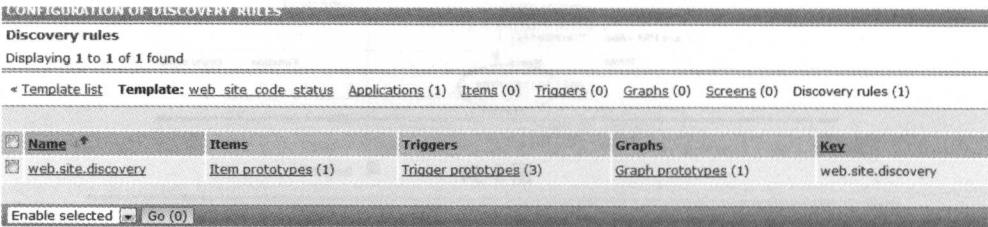


图 9-58

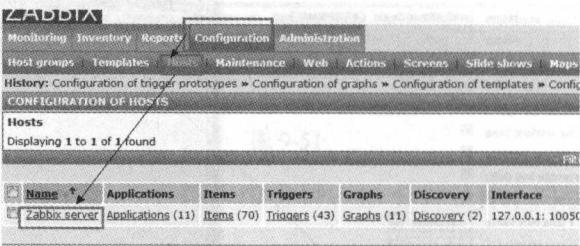


图 9-59

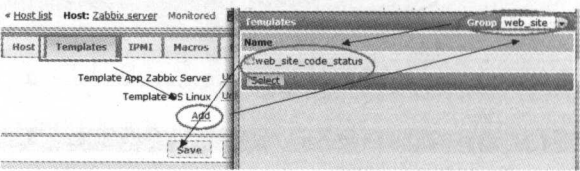


图 9-60

查看模板，如图 9-61 所示，已经成功添加了该模板。

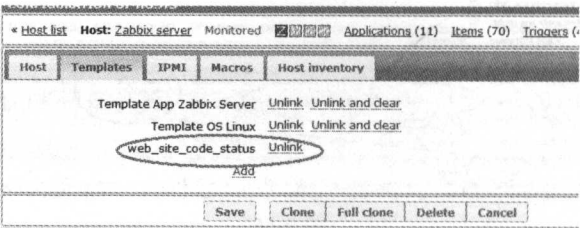


图 9-61

大约等 30 秒后，就可以看到刚才添加的三个站点监控，如图 9-62 和图 9-63 所示。

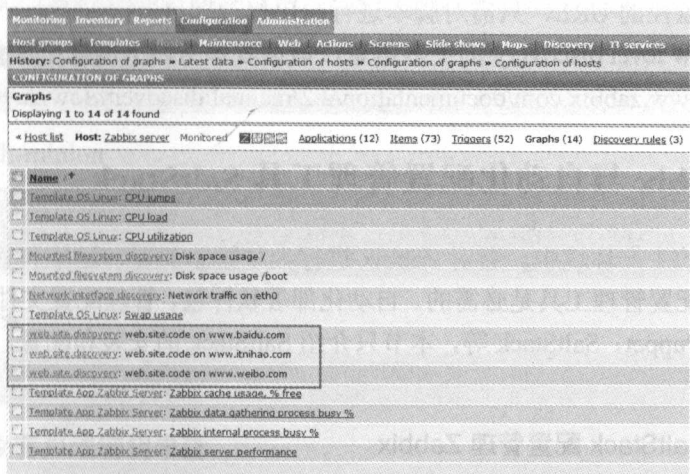


图 9-62

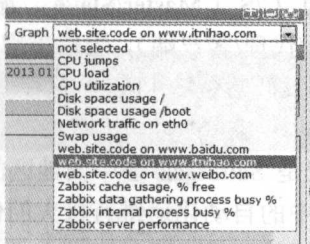


图 9-63

查看图形数据，如图 9-64 所示。

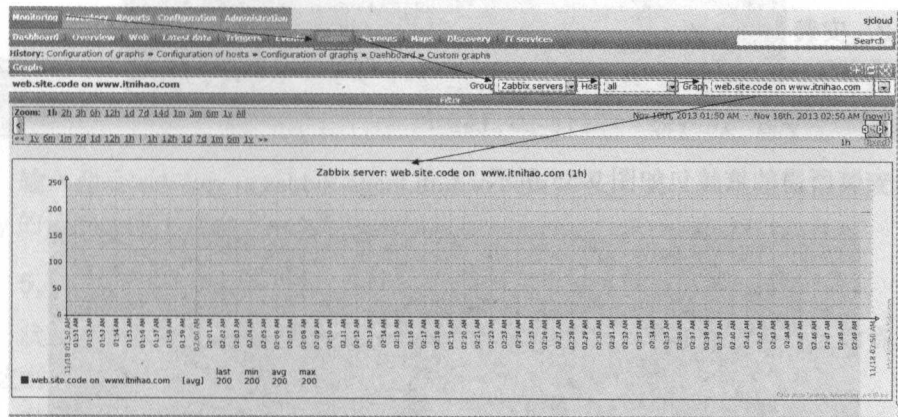


图 9-64

至此，前面的任务已经完成，如果我们对 web.txt 文件增加 URL，新的 URL 会自动添加到监控。举一反三，同样可以把 URL 存储到 CMDB 中，通过网络路径获取需要监控的 URL，只需对脚本进行简单修改即可。

关于 Low level discovery 的更多信息，请读者参考以下网址中的内容。

https://www.zabbix.com/documentation/2.2/manual/discovery/low_level_discovery

9.5 Zabbix 与自动化配置管理工具 SaltStack

在实际的生产环境中，会对 Zabbix 进行大规模的部署、运维和管理，此时，一套集中的配置管理工具是必需的。自动化部署软件包、管理配置文件等开源的工具具有 Chef、Puppet、SaltStack 等，本节只介绍 SaltStack 的安装和配置，其他工具与之类似。

1. 用 SaltStack 配置管理 Zabbix

SaltStack 是一个管理配置工具，其作用是为系统管理人员提供标准化的配置管理和命令执行，架构为 C/S（Master/Slave，服务器/客户端）或 C/P/S（Master/Poryx/Slave，服务器/代理/客户端），通信采用证书认证方式，开发语言为 Python，提供 API，二次开发较容易，可以运行在多种平台上，其官方网址为 <http://www.saltstack.com>。中国 SaltStack 用户组由@绿小小肥建立，网址为 <http://www.saltstack.cn>。绿肥是 SaltStack 在中国地区的布道者，对推进 SaltStack 在中国地区的发展和在企业中的自动化运维有极大的促进作用。

注意，一般采用自动化运维工具进行的配置管理，其软件安装都应该采用标准化安装，如 RPM 包安装，需要有软件仓库。因此，前提是将 Zabbix 放于指定的软件仓库中，定制 RPM 包，相关内容可参考第 15 章。

2. 安装 salt-master

```
shell# rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
shell# yum install salt-master
```

安装所需的依赖包如图 9-65 所示。

Installing:		
salt-master	noarch	2014.1.0-1.el6
salt-minion	noarch	2014.1.0-1.el6
Installing for dependencies:		
PyYAML	x86_64	3.10-3.el6
libyaml	x86_64	0.1.3-1.el6
openpgm	x86_64	5.1.118-3.el6
python-babel	noarch	0.9.4-5.1.el6
python-jinja2	x86_64	2.2.1-1.el6
python-msgpack	x86_64	0.1.13-3.el6
python-zmq	x86_64	2.2.0.1-1.el6
salt	noarch	2014.1.0-1.el6
sshpas	x86_64	1.05-1.el6
zeromq3	x86_64	3.2.4-1.el6

图 9-65

启动服务

```
shell# service salt-master start
shell# chkconfig salt-master on
```

3. 安装 salt-minion

```
shell# yum install salt-minion
```

配置 salt-minion

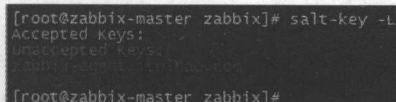
```
shell# vim /etc/salt/minion
master: salt-master.itnihao.com      #master的IP或域名
id: zabbix-agent.itnihao.com        #minion的标示
```

启动服务

```
shell# service salt-minion start
shell# chkconfig salt-minion on
```

4. 接收客户端密钥申请

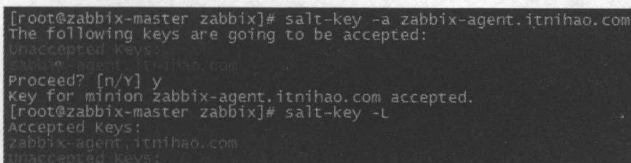
在 master 中配置: 查看客户端申请的 Key 请求, 命令为 salt-key -L, 如图 9-66 所示。



```
[root@zabbix-master zabbix]# salt-key -L
Accepted Keys:
Unaccepted Keys:
zabbix-agent.itnihao.com
[root@zabbix-master zabbix]#
```

图 9-66

接受客户端的 Key 请求, 如图 9-67 所示。



```
[root@zabbix-master zabbix]# salt-key -a zabbix-agent.itnihao.com
The following keys are going to be accepted:
Unaccepted Keys:
zabbix-agent.itnihao.com
Proceed? [n/Y] y
Key for minion zabbix-agent.itnihao.com accepted.
[root@zabbix-master zabbix]# salt-key -L
Accepted Keys:
zabbix-agent.itnihao.com
Unaccepted Keys:
```

图 9-67

输入命令 salt-key -a zabbix-agent.itnihao.com 后按 y 键 (如果是批量接受所有 Key 的请求, 可以用命令 salt-key -A)。

5. 状态同步文件

状态同步是指将定义好的配置信息同步到客户端主机中, 使客户端的资源状态达到目标状态。

在 salt-master 中的配置语句如下。

```
shell# mkdir /srv/salt/ #因salt安装好后无/srv/salt目录, 所以建立此文件夹
shell# vim /srv/salt/top.sls
base:
  '*.*itnihao.com':
    - zabbix
shell# mkdir /srv/salt/zabbix
shell# vim /srv/salt/zabbix/init.sls
zabbix-agent:
  service:
    - running
    - watch:
      - file: zabbix_agentd.conf
      - file: zabbix_agentd.conf.d
      - pkg: zabbix-agentd
  require:
    - pkg: zabbix-agent
zabbix-agentd:
  pkg.installed:
    - name: zabbix-agent
    - version: '2.2.2-0.el6.zbx'
    - skip_verify: True
    - skip_suggestions: True
    - fromrepo: zabbix
    - refresh: True

zabbix_agentd.conf:
  file.managed:
    - name: /etc/zabbix/zabbix_agentd.conf
    - source: salt://zabbix/conf/zabbix_agentd.conf
    - mode: 644
    - user: zabbix
    - group: zabbix
    - template: jinja
zabbix_agentd.conf.d:
  file.recurse:
    - name: /etc/zabbix/zabbix_agentd.conf.d
    - source: salt://zabbix/conf/zabbix_agentd.conf.d
    - include_empty: True
    - user: zabbix
    - group: zabbix
    - dir_mode: 755
    - file_mode: 644
scripts:
  file.recurse:
    - name: /etc/zabbix/scripts
    - source: salt://zabbix/scripts
    - include_empty: True
    - user: zabbix
    - group: zabbix
    - dir_mode: 755
    - file_mode: 700
```

整个代码的结构目录如下。

```

/srv/salt/
├── top.sls
└── zabbix
    ├── conf
    │   ├── zabbix_agentd.conf      #zabbix-agent的配置文件
    │   └── zabbix_agentd.conf.d    #子配置文件
    │       ├── haproxy_host_status.conf
    │       ├── haproxy_main_status.conf
    │       └── haproxy_status_discovery.conf
    ├── init.sls                    #salt的状态配置文件
    └── scripts
        ├── haproxy_host_status
        ├── haproxy_main_status
        └── haproxy_status_discovery

```

6. 执行状态同步

```
shell# salt '*' state.highstate
```

执行结果如图 9-68 所示。

```

[root@zabbix-master salt]# salt '*' state.highstate
zabbix-agentd:1917466.com
-----
ID: zabbix_agentd.conf
Function: file.managed
Name: /etc/zabbix/zabbix_agentd.conf
Result: True
Comment: file /etc/zabbix/zabbix_agentd.conf is in the correct state
Changes:
-----
ID: zabbix_agentd.conf.d
Function: file.recurse
Name: /etc/zabbix/zabbix_agentd.conf.d
Result: True
Comment: The directory /etc/zabbix/zabbix_agentd.conf.d is in the correct state
Changes:
-----
ID: zabbix_agentd
Function: pkg.installed
Name: zabbix-agent
Result: True
Comment: version 3.2.2-6.el6.x86_64 of package 'zabbix-agent' is already installed
Changes:
-----
ID: zabbix-agent
Function: service.running
Result: True
Comment: the service zabbix-agent is already running
Changes:
-----
ID: scripts
Function: file.recurse
Name: /etc/zabbix/scripts
Result: True
Comment: The directory /etc/zabbix/scripts is in the correct state
Changes:
-----
Summary
-----
Succeeded: 7
Failed: 0
-----

```

图 9-68

关于 SaltStack 更多的功能，请读者自行进行深入学习。

第 10 章 使用的经验和技巧

在 Zabbix 的配置使用中，有很多需要注意的地方，因此，这部分内容将单独整理为一章进行介绍，方便读者查阅。

10.1 如何有效地设置监控告警

对于监控的告警信息，若处理得好，将会提高故障响应速度，反之，则会影响我们的工作情绪，进而影响工作效率。试想，若一天收到 1000 封告警信息，是否还会去逐一查看监控告警信息？是否还能分辨是重大故障，还是一般故障？

对于误报、漏报的情况，会让人对信息的警觉性放松，时间久了，还会导致对接收的监控信息有反感。所以，对于监控告警信息的发送，是一件特别慎重的事情。下面总结一下对监控告警信息的需求。

- **基于业务类型**：将告警信息发送给相应的业务用户，例如，IDC 人员、Web 运维人员、CDN 运维人员、网络运维人员。不同的人员管理不同的设备，因此，需要把故障发送给相关用户进行处理。
- **基于故障级别**：对一个故障，将不同的故障级别发送给不同的用户，例如，5 分钟内的故障发送给运维一线人员，10 分钟的故障发送给运维部门主管，30 分钟的故障发送给运维部门经理。特大故障发送给部门相关领导。
- **基于时间发送**：比如业务维护期，告警无须发送。
- **故障的相关依赖关系**：当 A 服务发生故障时，发送一般告警；当 A、B 服务发生故障时，发送业务故障告警。对出现故障的服务尝试用相关命令或者脚本进行操作处理，尝试自动恢复，例如，重启服务、重启服务器等。

告警信息需要定制，重点关注以下内容。

- 哪些业务需要告警？
- 哪种故障需要告警？
- 告警等级如何划分？
- 故障依赖关系如何定义？
- 告警信息如何汇集？
- 如何做到精准有效地告警？

1. 基于业务类型

将不同的用户分配到不同的用户组。Action 中定义不同主机组发送信息给对应的用户组；Action 中可以为不同的服务器组创建不同的 Action，因此，可以基于对业务类型的机器分组进行告警信息的发送。

如图 10-1 到图 10-5 所示，是基于组发送的一个告警配置。

CONFIGURATION OF ACTIONS

Action Conditions Operations

Name: mail_to_user_group1

Default operation step duration: 60 (minimum 60 seconds)

Default subject: group1服务器发生故障 {TRIGGER.STATUS}: {TI}

Default message: Trigger: {TRIGGER.NAME}
Trigger status: {TRIGGER.STATUS}
Trigger severity: {TRIGGER.SEVERITY}
Trigger URL: {TRIGGER.URL}
Trigger DATE: {EVENT.DATE}
Trigger Time: {TIME}
Item values:

Recovery message: ☒

Recovery subject: group1服务器故障已经解决 {TRIGGER.STATUS}: {TI}

Recovery message: Trigger: {TRIGGER.NAME}
Trigger status: {TRIGGER.STATUS}
Trigger severity: {TRIGGER.SEVERITY}
Trigger URL: {TRIGGER.URL}
Trigger DATE: {EVENT.DATE}
Trigger Time: {TIME}
Item values:

Enabled: ☒

Save Clone Delete Cancel

图 10-1

CONFIGURATION OF ACTIONS

Action Conditions Operations

Type of calculation: AND (A) and (B) and (C)

Conditions:

Label	Name	Action
(A)	Maintenance status not in "maintenance"	Remove
(B)	Trigger value = "PROBLEM"	Remove
(C)	Host group = "group1"	Remove

New condition: Trigger name like

Add

Save Clone Delete Cancel

图 10-2

Operations

Action operations

Steps	Details
1	Send message to us
	New

图 10-3

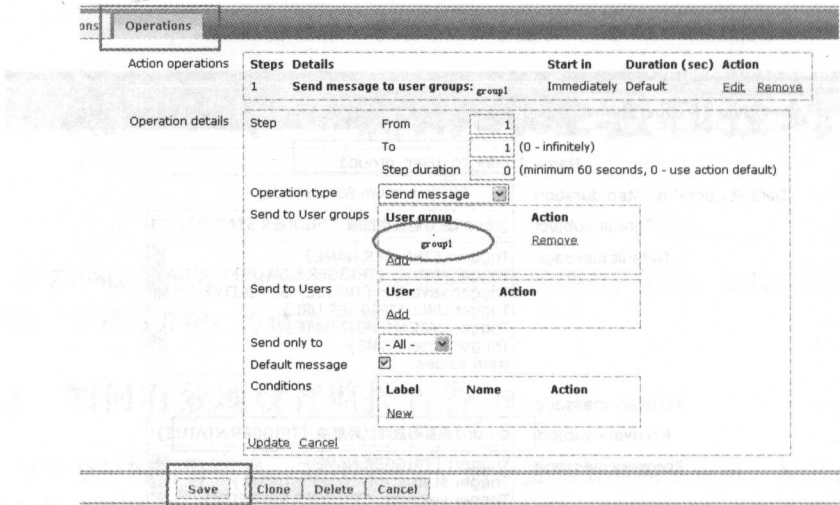


图 10-4

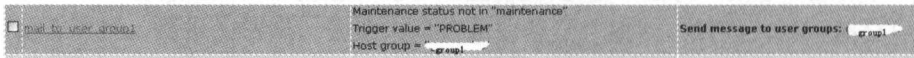


图 10-5

当然，用户组中的用户需要在用户管理中定义，此处略（有关用户管理的内容，在此省略）。

2. 基于故障级别

在定义触发器的时候，有故障级别的选择，如图 10-6 所示。

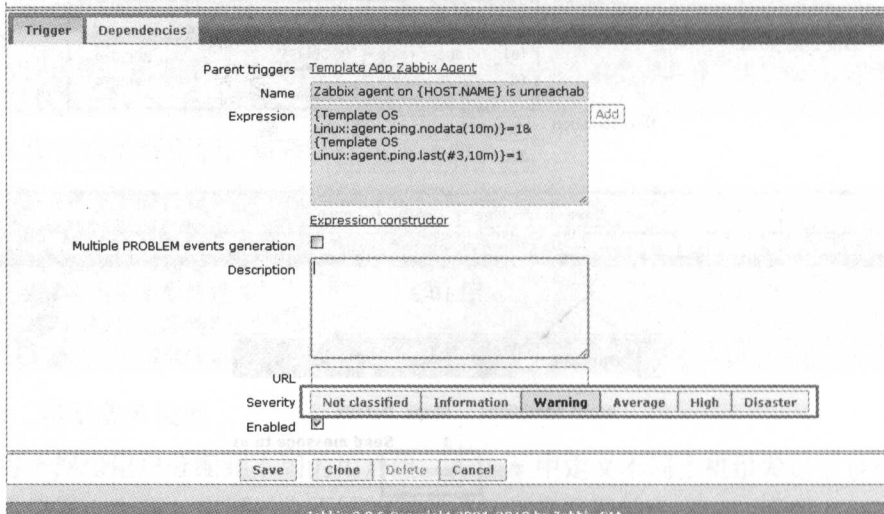


图 10-6

我们可以对不同的故障分级别定义。

在对用户添加告警媒体的时候，可以定义不同的故障等级发送。例如，触发器如下。

```
{Template OS Linux:agent.ping.nodata(5m)}=1&{Template OS Linux:agent.ping.last(#3,5m)}=1
{Template OS Linux:agent.ping.nodata(10m)}=1&{Template OS Linux:agent.ping.last(#3,10m)}=1
{Template OS Linux:agent.ping.nodata(15m)}=1&{Template OS Linux:agent.ping.last(#3,15m)}=1
{Template OS Linux:agent.ping.nodata(30m)}=1&{Template OS Linux:agent.ping.last(#3,30m)}=1
```

然后将不同的触发器标记为不同的故障级别，如图 10-7 所示。

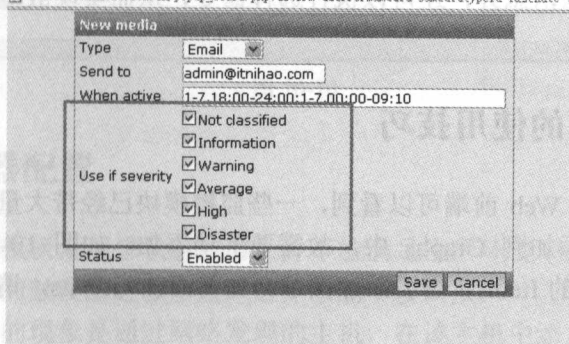


图 10-7

基于以上两个功能，完全可以做到将不同的故障级别发送给不同的人。

3. 基于时间发送

如图 10-8 所示，时间可以为 1-7,00:00-24:00、1-7,00:00-09:10 等，各时间段之间用分号隔开，可以创建任意的时间段，根据实际情况选择即可。

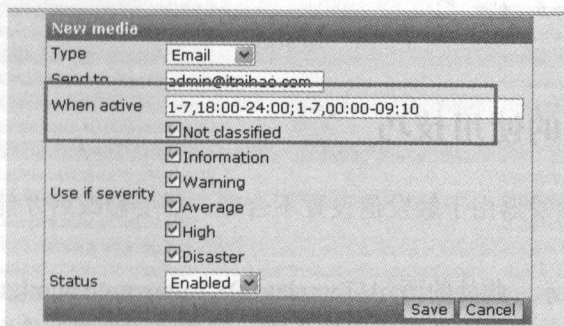


图 10-8

4. 故障依赖关系

```
{Template OS Linux:agent.ping.nodata(5m)}=1&{Template OS Linux:agent.ping.last(#3,5m)}=1
```

以上触发器的设置条件语句中，表示 Agent 在 5 分钟内无数据，并且在 5 分钟内的第三次取值是有的，即前 2 分钟设备还存活。

除此之外，在这种环境下：一个区域网络断开，按照常规的告警方式，会发送该网段所有设备的告警，但通过只发送一个告警。也可以配置告警的故障依赖。

5. 故障的自动恢复

告警除了发送消息外，还可以执行远程命令，从而实现故障的自动恢复，具体内容见 6.3.6 节。

10.2 监控项的使用技巧

在 Zabbix 的 Web 前端可以看到，一些监控模块已经带大量的监控项，但某些监控项并没有添加到 Graphs 中，故需要手动添加。如图 10-9 所示，Key 为 system.users.num 的 Items，需要添加图形（如何添加 Graphs，请参考前面章节的内容）。

The screenshot shows the 'Add new item' form in the Zabbix web interface. The fields are as follows:

- Host: Template OS Linux
- Name: Number of logged in users
- Type: Zabbix agent (dropdown menu)
- Key: system.users.num (with a 'Select' button)
- Data type: Numeric (unsigned) (dropdown menu)
- Units: Decimal (dropdown menu)
- Multiplier: 1 (checkbox and input field)

图 10-9

10.3 触发器的使用技巧

默认的一些触发器由于触发值设置不合理，需要修改后才能满足自己的生产环境。

如图 10-10 所示，此处的语句 {Template OS Linux:proc.num[,run].last(0)}>30 表示大于 30 即告警。由于服务器应用的不同，需要将此值进行合理设置，否则会产生大量的误报。

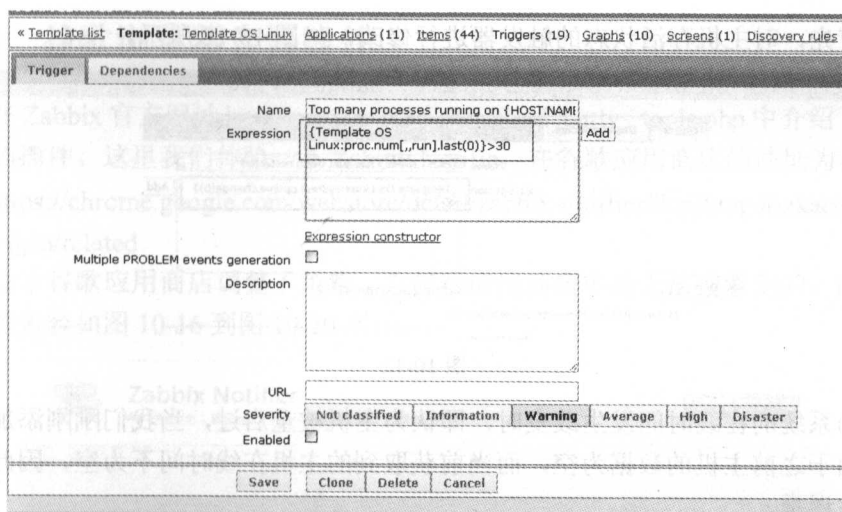


图 10-10

10.4 触发器配置

由于默认的模板是理想状态的值，并不能满足生产环境中各种奇特的需求，所以需要对默认的触发器进行修改，这样才能不被误报。

一个很常见的现象是通过网络发现的主机，在该主机中添加机器的时候，会出现告警提示机器刚刚发生过重启，这种情况就属于严重的误报，且随着主机规模的增大，这种故障会很严重。

如图 10-11 所示为 Linux 的模板，默认有如下的触发器设置，应根据实际需求进行修改。

Severty	Name	Expression
Warning	/etc/passwd has been changed on {HOST.NAME}	{Template OS Linux:yfs.file.chsumf/e
Information	Configured max number of opened files is too low on {HOST.NAME}	{Template OS Linux:kernel.maxfiles.l
Information	Configured max number of processes is too low on {HOST.NAME}	{Template OS Linux:kernel.maxproc
Warning	Disk I/O is overloaded on {HOST.NAME}	{Template OS Linux:system.cpu.iutil
Information	Host information was changed on {HOST.NAME}	{Template OS Linux:system.uname.e
Information	Template App Zabbix Agent: Host name of zabbix_agentd was changed on {HOST.NAME}	{Template OS Linux:agent.hostname
Information	Hostname was changed on {HOST.NAME}	{Template OS Linux:system.hostname
Average	Lack of available memory on server {HOST.NAME}	{Template OS Linux:vm.memory.size
Warning	Lack of free swap space on {HOST.NAME}	{Template OS Linux:system.swap.sp
Warning	Processor load is too high on {HOST.NAME}	{Template OS Linux:system.cpu.load
Warning	Too many processes on {HOST.NAME}	{Template OS Linux:proc.num[.avg(
Warning	Too many processes running on {HOST.NAME}	{Template OS Linux:proc.num[.run]
Information	Template App Zabbix Agent: Version of zabbix_agentd was changed on {HOST.NAME}	{Template OS Linux:agent.version.d
Average	Template App Zabbix Agent: Zabbix agent on {HOST.NAME} is unreachable for 5 minutes	{Template OS Linux:agent.ping.no
Information	{HOST.NAME} has just been restarted	{Template OS Linux:system.uptime.c Linux:agent.pino.last(#1.5m)}=1

图 10-11

首先，对主机存活状态的触发器进行修改，如图 10-12 所示。

```
{Template OS Linux:system.uptime.change(0)}<0
```

Trigger Dependencies

Name: {HOST.NAME} has just been restarted

Expression: {Template OS Linux:system.uptime.change(0)}<0

Multiple PROBLEM events generation: ☐

Description:

图 10-12

当系统的在线时间发生改变时，即认为主机被重启过，当我们刚刚添加完主机，由于之前主机的数据为空，而当前获取到的主机在线时间不为空，因此，就会发生误报。

将触发器改为如图 10-13 所示的结果，意思是启动时间发生改变，且 5 分钟内的第一次检测 Agent 是存活的，即 5 分钟前系统也是启动的，随即触发事件，从而引发后面的告警。

```
{Template OS Linux:system.uptime.change(0)}<0&{Template OS Linux:agent.ping.last(#1,5m)}=1
```

Trigger Dependencies

Name: {HOST.NAME} has just been restarted

Expression: {Template OS Linux:system.uptime.change(0)}<0&{Template OS Linux:agent.ping.last(#1,5m)}=1

Multiple PROBLEM events generation: ☐

Description:

图 10-13

然后，对文件描述符进行修改，如图 10-14、图 10-15 所示。

Name: Too many processes on {HOST.NAME}

Expression: {Template OS Linux:proc.num[.].avg(5m)}>300

图 10-14

Name: Too many processes running on {HOST.NAME}

Expression: {Template OS Linux:proc.num[.run].avg(5m)}>30

图 10-15

注意：在 16.4.3 节还列举了其他不合理的触发器设置，读者可以参考。

10.5 谷歌浏览器告警插件

在 Zabbix 官方网址 http://www.zabbix.com/third_party_tools.php 中介绍了很多有趣的插件。这里我们体验一下 Zabbix Notifie，在谷歌应用商店的地址为：

<https://chrome.google.com/webstore/detail/zabbix-notifier/ikeijbnpddnkaeejokgifiocbcijjfo/related>

由于谷歌应用商店调整了策略，直接在应用商店中是无法搜索到的。配置该插件的内容如图 10-16 到图 10-20 所示。

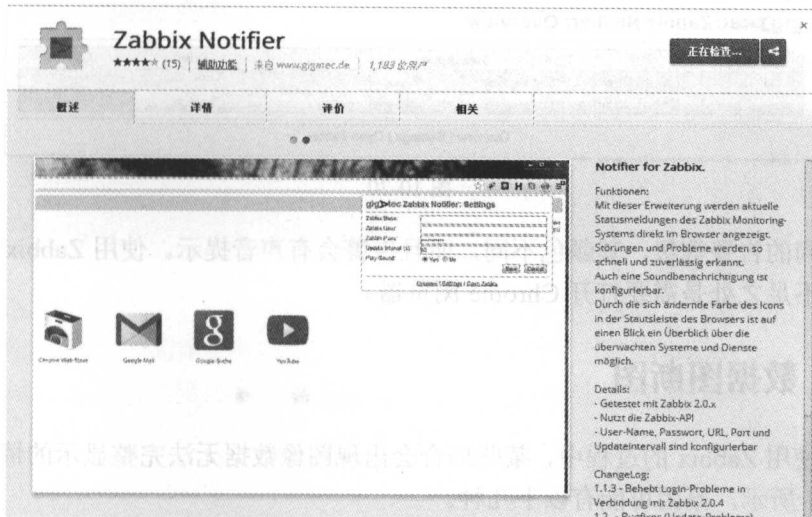


图 10-16

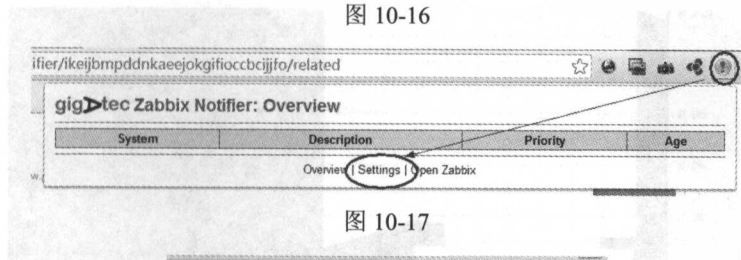


图 10-17

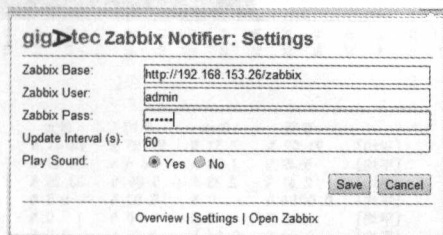


图 10-18

注：图 10-18 的 Zabbix Base 文本框中的内容为访问的路径。

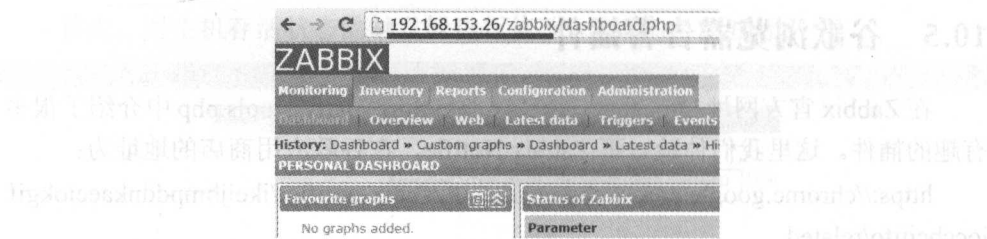


图 10-19

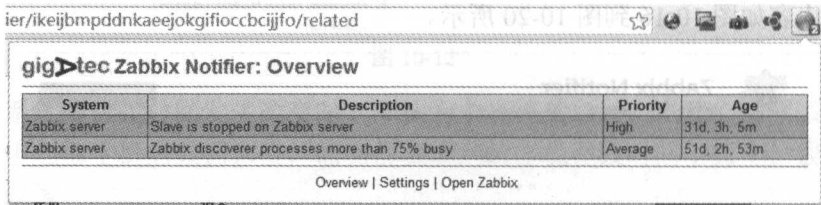


图 10-20

不同的告警等级，其颜色不同，而且告警会有声音提示。使用 Zabbix Notifier 插件的不足之处是需要打开 Chrome 浏览器。

10.6 数据图断图

在使用 Zabbix 的过程中，某些场合会出现图像数据无法完整显示的情况，如图 10-21 所示，原因可能有以下几种。

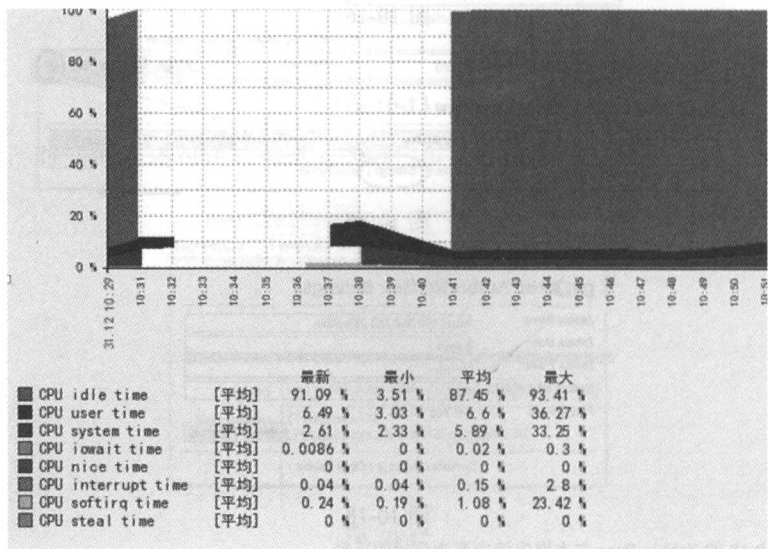


图 10-21

- 系统负载高。
- TCP连接数过多。
- 网络问题。
- 无法获取到数据。
- 数据库无法存储数据。

对这些故障的处理方法是对系统进行调优。有关的内容请读者参考第 12 章。

第 11 章 监控案例

本章将介绍 Zabbix 监控的大量实例，通过这些实例的学习，读者可以更深入地理解 Zabbix 强大的自定义功能。希望读者通过本章的学习，做到举一反三，能更好地应用 Zabbix 来监控自己的业务。

11.1 监控 TCP 连接数

在服务器的性能监控中，有一项很重要的指标，即 TCP 的连接状态。从 TCP 的连接状态中可以看到网络的连接情况、服务器的压力情况，对服务器的并发有一个很好的直观反应，所以监控显得特别重要。

对 TCP 的监控可以采用 ss、netstat、/proc/net/tcp 这三个不同的方法来实现，从性能上说，ss 是最快的，按照实际经验，netstat 在并发数高于 2 万个的情况下会出现严重的卡顿现象，如果用 netstat 来采集数据，会出现在高并发的情况下无法采集到数据的情况，从而使监控造成数据无法采集的情况出现。

(1) ss 命令

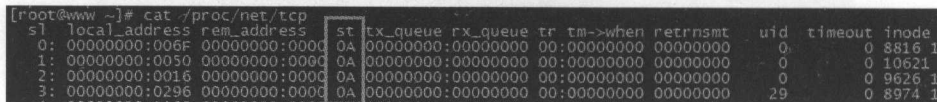
```
shell# ss state all|awk '{++S[$1]} END {for(a in S) {printf "%11-s %s\n", a,S[a]}}'
```

(2) netstat 命令

```
shell# netstat -an|awk '/^tcp/ {++S[$NF]} END {for(a in S) {printf "%11-s %s\n", a,S[a]}}'
```

(3) /proc/net/tcp 文件

监控 TCP 连接状态，可读取 /proc/net/tcp 文件，其中第四列为 TCP 连接各个状态，如图 11-1 所示。



sl	local_address	rem_address	st	tx_queue	rx_queue	tr	tm->when	retransmt	uid	timeout	inode
0:	00000000:006F	00000000:0000	0A	00000000:00000000	00:00000000	00000000	0	0	8816	1	
1:	00000000:0050	00000000:0000	0A	00000000:00000000	00:00000000	00000000	0	0	10621		
2:	00000000:0016	00000000:0000	0A	00000000:00000000	00:00000000	00000000	0	0	9626	1	
3:	00000000:0296	00000000:0000	0A	00000000:00000000	00:00000000	00000000	29	0	8974	1	

图 11-1

各状态代表的意思如下。

```
00 "ERROR_STATUS",
01 "TCP_ESTABLISHED",
```

```

02 "TCP_SYN_SENT",
03 "TCP_SYN_RECV",
04 "TCP_FIN_WAIT1",
05 "TCP_FIN_WAIT2",
06 "TCP_TIME_WAIT",
07 "TCP_CLOSE",
08 "TCP_CLOSE_WAIT",
09 "TCP_LAST_ACK",
0A "TCP_LISTEN",
0B "TCP_CLOSING"

```

(4) 各种方法的性能比较

netstat 运行的时间为 0.398s, 如图 11-2 所示。

```

[root@ ~]# time netstat -an|awk '/^tcp/ {++S[$NF]} END {for(a in S) {printf "%11-s %s\n", a,S[a]}}'
TIME_WAIT      2
CLOSE_WAIT     11
ESTABLISHED    2400
SYN_RECV       1
LAST_ACK       1
LISTEN         34

real    0m0.398s
user    0m0.061s
sys     0m0.367s

```

图 11-2

ss 运行的时间为 0.035s, 如图 11-3 所示。

```

[root@ ~]# time ss state all|awk '++S[$1] END {for(a in S) {printf "%11-s %s\n", a,S[a]}}'
LAST-ACK       1
SYN-RCV        1
ESTAB          2405
State          1
TIME-WAIT      2
CLOSE-WAIT     11
LISTEN         34

real    0m0.035s
user    0m0.014s
sys     0m0.020s

```

图 11-3

/proc/net/tcp 运行的时间为 4.034s, 最慢, 如图 11-4 所示。

```

[root@ ~]# time bash tcp.sh
0
2401
0
4
0
0
2
0
0
0
8
1
26

real    0m4.034s
user    0m0.059s
sys     0m3.989s

```

图 11-4

当然，以上的测试与使用 `awk` 有关系。另外，还与并发数量有很大关系，当前并发数为 2400 个左右。

这里以 `/proc/net/tcp` 为例，展示如何监控 TCP 的连接数（如果使用 `ss` 命令，其实现过程与之类似）。

```
shell# vim /etc/zabbix/zabbix_agentd.conf.d/tcp_connect.conf
userparameter=tcp_connect.errorstatus,awk '($4 == "00") {print $4}'
'/proc/net/tcp|awk '{++S[$NF]} END {for(a in S) {printf "%s", S[a]}}'
userparameter=tcp_connect.established,awk '($4 == "01") {print $4}'
'/proc/net/tcp|awk '{++S[$NF]} END {for(a in S) {printf "%s", S[a]}}'
userparameter=tcp_connect.synsent,awk '($4 == "02") {print $4}'
'/proc/net/tcp|awk '{++S[$NF]} END {for(a in S) {printf "%s", S[a]}}'
userparameter=tcp_connect.synrecv,awk '($4 == "03") {print $4}'
'/proc/net/tcp|awk '{++S[$NF]} END {for(a in S) {printf "%s", S[a]}}'
userparameter=tcp_connect.finwait1,awk '($4 == "04") {print $4}'
'/proc/net/tcp|awk '{++S[$NF]} END {for(a in S) {printf "%s", S[a]}}'
userparameter=tcp_connect.finwait2,awk '($4 == "05") {print $4}'
'/proc/net/tcp|awk '{++S[$NF]} END {for(a in S) {printf "%s", S[a]}}'
userparameter=tcp_connect.timewait,awk '($4 == "06") {print $4}'
'/proc/net/tcp|awk '{++S[$NF]} END {for(a in S) {printf "%s", S[a]}}'
userparameter=tcp_connect.close,awk '($4 == "07") {print $4}'
'/proc/net/tcp|awk '{++S[$NF]} END {for(a in S) {printf "%s", S[a]}}'
userparameter=tcp_connect.closing,awk '($4 == "0B") {print $4}'
'/proc/net/tcp|awk '{++S[$NF]} END {for(a in S) {printf "%s", S[a]}}'
userparameter=tcp_connect.close_wait,awk '($4 == "08") {print $4}'
'/proc/net/tcp|awk '{++S[$NF]} END {for(a in S) {printf "%s", S[a]}}'
userparameter=tcp_connect.last_ack,awk '($4 == "09") {print $4}'
'/proc/net/tcp|awk '{++S[$NF]} END {for(a in S) {printf "%s", S[a]}}'
userparameter=tcp_connect.listen,awk '($4 == "0A") {print $4}'
'/proc/net/tcp|awk '{++S[$NF]} END {for(a in S) {printf "%s", S[a]}}'
```

在 Web 界面中添加 Key，如图 11-5 所示。

Template list Template: Template App Top Connect Applications (1) Items (9) Triggers (0) Graphs (1)				
Wizard	Name *	Triggers	Key	Interval
	tcp_connect.closewait		tcp_connect.closewait	120
	tcp_connect.established		tcp_connect.established	120
	tcp_connect.finwait1		tcp_connect.finwait1	120
	tcp_connect.finwait2		tcp_connect.finwait2	120
	tcp_connect.listen		tcp_connect.listen	120
	tcp_connect.synrecv		tcp_connect.synrecv	120
	tcp_connect.synsent		tcp_connect.synsent	120
	tcp_connect.timeclose		tcp_connect.timeclose	120
	tcp_connect.timewait		tcp_connect.timewait	120

图 11-5

11.2 监控 Nginx

在 `nginx.conf` 中添加如下内容。


```
server {
    listen 127.0.0.1:80;
    server_name 127.0.0.1;
    location /nginxstatus {
        stub_status on;
        access_log off;
        allow 127.0.0.1;
        allow 192.168.11.0/24;    #这里是允许的IP地址
        deny all;
    }
}
```

如果 Nginx 通过 Puppet、SaltStack 等配置管理工具进行统一管理，则需要在模板中统一加上这段配置。

关于 Nginx 状态的更多解释，请读者参考以下网址中的内容。

http://nginx.org/en/docs/http/nginx_http_status_module.html#status

监控 Nginx 的脚本语句如下。

```
shell# vim /etc/zabbix/scripts/nginx_status
#!/bin/bash

# function:monitor nginx from zabbix
# License: GPL
# mail:itnihao@qq.com
# version 1.0 date:2012-12-09
# version 1.0 date:2013-01-15
# Functions to return nginx stats
function active {
    curl "http://127.0.0.1/nginxstatus" | awk '/Active/ {print $3}'
}
function reading {
    curl "http://127.0.0.12/nginxstatus" | awk '/Reading/ {print $2}'
}
function writing {
    curl "http://127.0.0.1/nginxstatus" | awk '/Writing/ {print $4}'
}
function waiting {
    curl "http://127.0.0.1/nginxstatus" | awk '/Waiting/ {print $6}'
}
function accepts {
    curl "http://127.0.0.1/nginxstatus" | awk NR==3 | awk '{print $1}'
}
function handled {
    curl "http://127.0.0.1/nginxstatus" | awk NR==3 | awk '{print $2}'
}
function requests {
```



```

curl "http://127.0.0.1/nginxstatus" | awk NR==3 |
awk '{print $3}'
}

case "$1" in
active)
    active
    ;;
reading)
    reading
    ;;
writing)
    writing
    ;;
waiting)
    waiting
    ;;
accepts)
    accepts
    ;;
handled)
    handled
    ;;
requests)
    requests
    ;;
*)
    echo "Usage: $0 {nginx_site_discovery}"
    echo "Usage: $0 {active [host]|reading [host]|writing [host]|waiting [host]|accepts [host]|handled [host]|requests [host]}"
esac

```

Key 的配置文件如下:

```

shell# vim /etc/zabbix/zabbix_agentd.conf.d/monitor_nginx.conf
UserParameter=nginx.accepts,/etc/zabbix/scripts/nginx_status accepts
UserParameter=nginx.handled,/etc/zabbix/scripts/nginx_status handled
UserParameter=nginx.requests,/etc/zabbix/scripts/nginx_status requests
UserParameter=nginx.connections.active,/etc/zabbix/scripts/nginx_status active
UserParameter=nginx.connections.reading,/etc/zabbix/scripts/nginx_status reading
UserParameter=nginx.connections.writing,/etc/zabbix/scripts/nginx_status writing
UserParameter=nginx.connections.waiting,/etc/zabbix/scripts/nginx_status waiting

```

11.3 监控 PHP-FPM

PHP-FPM 工作模式通常与 Nginx 结合使用。

修改/etc/php-fpm.conf 的语句如下。

```
shell# vim /etc/php-fpm.conf
pm.status_path = /phpfpmstatus
shell# /etc/init.d/php-fpm restart
```

修改/etc/nginx/nginx.conf 的配置文件, 通过 Nginx 访问 PHP-FPM 的状态。

```
server {
    listen 127.0.0.1:80;
    server_name 127.0.0.1;
    location /nginxstatus {
        stub_status on;
        access_log off;
        allow 127.0.0.1;
        deny all;
    }
    location ~ ^/(phpfpmstatus)$ {
        include fastcgi_params;
        fastcgi_pass unix:/tmp/fpm.sock;
        fastcgi_param SCRIPT_FILENAME $fastcgi_script_name;
    }
}
```

PHP-FPM 监控的脚本语句如下:

```
shell# vim /etc/zabbix/scripts/monitor_phpfpmp_status
#!/bin/bash

# function:monitor php-fpm status from zabbix
# License: GPL
# mail:itnihao@qq.com
# date:2013-05-30

source /etc/bashrc >/dev/null 2>&1
source /etc/profile >/dev/null 2>&1

LOG_FILE=/var/log/zabbix/phpfpmstatus.log
curl http://127.0.0.1/phpfpmstatus >${LOG_FILE}

pool(){
    awk '/pool/ {print $NF}' ${LOG_FILE}
}
process_manager() {
    awk '/process manager/ {print $NF}' ${LOG_FILE}
}

start_since(){
    awk '/start since:/ {print $NF}' ${LOG_FILE}
}
accepted_conn(){
    awk '/accepted conn:/ {print $NF}' ${LOG_FILE}
}
listen_queue(){
    awk '/listen queue:/ {print $NF}' ${LOG_FILE}
}
max_listen_queue(){
```

```
        awk '/max listen queue:/ {print $NF}' ${LOG_FILE}
    }
    listen_queue_len(){
        awk '/listen queue len:/ {print $NF}' ${LOG_FILE}
    }
    idle_processes(){
        awk '/idle processes:/ {print $NF}' ${LOG_FILE}
    }
    active_processes(){
        awk '/active processes:/ {print $NF}' ${LOG_FILE}
    }
    total_processes(){
        awk '/total processes:/ {print $NF}' ${LOG_FILE}
    }
    max_active_processes(){
        awk '/max active processes:/ {print $NF}' ${LOG_FILE}
    }
    max_children_reached(){
        awk '/max children reached:/ {print $NF}' ${LOG_FILE}
    }
}

case "$1" in
pool)
    pool
    ;;
process_manager)
    process_manager
    ;;
start_since)
    start_since
    ;;
accepted_conn)
    accepted_conn
    ;;
listen_queue)
    listen_queue
    ;;
max_listen_queue)
    max_listen_queue
    ;;
listen_queue_len)
    listen_queue_len
    ;;
idle_processes)
    idle_processes
    ;;
active_processes)
    active_processes
    ;;
total_processes)
    total_processes
    ;;
max_active_processes)
    max_active_processes)
```



```

        max_active_processes
        ;;
    max_children_reached)
        max_children_reached
        ;;
*)
    echo "Usage: $0 {pool|process_manager|start_since|accepted_conn|listen_queue|max_listen_queue|listen_queue_len|idle_processes|active_processes|total_processes|max_active_processes|max_children_reached}"
esac

```

Key 的 php-fpm.conf 的子配置文件如下。

```

shell# cat /etc/zabbix/zabbix_agentd.conf.d/php-fpm.conf
UserParameter=phpfpm.status.pool,/etc/zabbix/scripts/monitor_php
fpm_status pool
UserParameter=phpfpm.status.process.manager,/etc/zabbix/scripts/
monitor_php fpm_status process_manager
UserParameter=phpfpm.status.start.since,/etc/zabbix/scripts/moni
tor_php fpm_status start_since
UserParameter=phpfpm.status.accepted.conn,/etc/zabbix/scripts/mo
nitor_php fpm_status accepted_conn
UserParameter=phpfpm.status.listen.queue,/etc/zabbix/scripts/mon
itor_php fpm_status listen_queue
UserParameter=phpfpm.status.max.listen.queue,/etc/zabbix/scripts
/monitor_php fpm_status max_listen_queue
UserParameter=phpfpm.status.listen.queue.len,/etc/zabbix/scripts
/monitor_php fpm_status listen_queue_len
UserParameter=phpfpm.status.idle.processes,/etc/zabbix/scripts/m
onitor_php fpm_status idle_processes
UserParameter=phpfpm.status.active.processes,/etc/zabbix/scripts
/monitor_php fpm_status active_processes
UserParameter=phpfpm.status.total.processes,/etc/zabbix/scripts/
monitor_php fpm_status total_processes
UserParameter=phpfpm.status.max.active.processes,/etc/zabbix/scr
ipts/monitor_php fpm_status max_active_processes
UserParameter=phpfpm.status.max.children.reached,/etc/zabbix/scr
ipts/monitor_php fpm_status max_children_reache

```

在 Web 页面中依次添加 Key，如图 11-6 所示。

zabbix	Name	Triggers	Key	Interval
	phpfpm.status.accepted.conn		phpfpm.status.accepted.conn	300
	phpfpm.status.active.processes		phpfpm.status.active.processes	300
	phpfpm.status.idle.processes		phpfpm.status.idle.processes	300
	phpfpm.status.listen.queue		phpfpm.status.listen.queue	300
	phpfpm.status.listen.queue.len		phpfpm.status.listen.queue.len	300
	phpfpm.status.max.active.processes		phpfpm.status.max.active.processes	300
	phpfpm.status.max.children.reached		phpfpm.status.max.children.reached	300
	phpfpm.status.max.listen.queue		phpfpm.status.max.listen.queue	300
	phpfpm.status.pool		phpfpm.status.pool	300
	phpfpm.status.process.manager		phpfpm.status.process.manager	300
	phpfpm.status.start.since		phpfpm.status.start.since	300
	phpfpm.status.total.processes		phpfpm.status.total.processes	300

图 11-6

监控效果如图 11-7 所示。

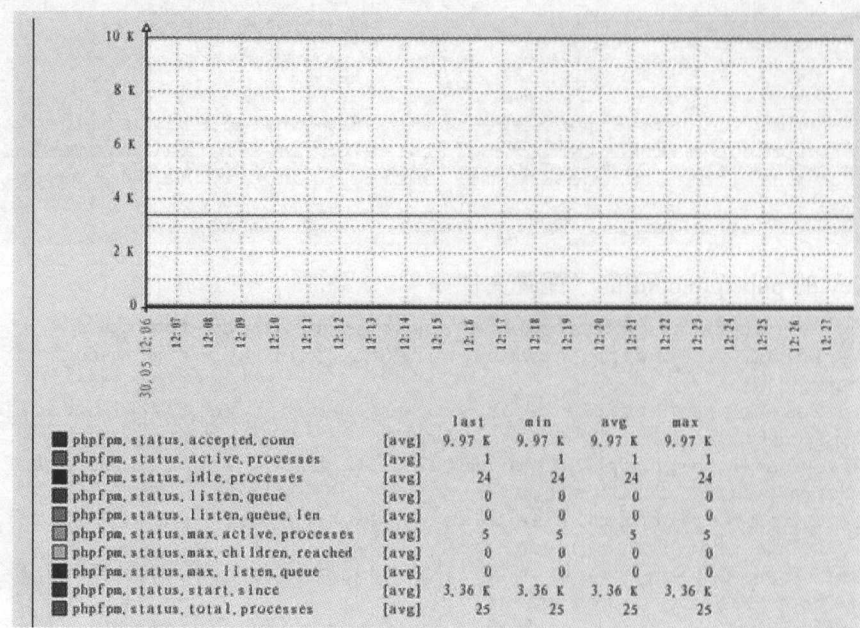


图 11-7

有关监控 Tomcat 和 Weblogic 的内容，请参考 7.7.7 节和 7.7.8 节的内容。

11.4 监控 MySQL

11.4.1 用自带的模板监控 MySQL

安装好 Zabbix 后，会出现一个 MySQL 的模板，如图 11-8 所示。

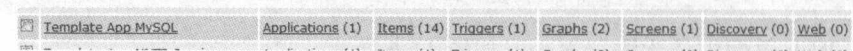


图 11-8

由于 Zabbix-Agent 本身并没有提供直接对 MySQL 监控的 Key，所以，首先需要通过自定义 Key 来应用这个模板。系统默认已经提供了如图 11-9 所示的模板。

接下来定义 Key。分析一下 Key 的规则可知，默认模板中，Key 可以归为以下三类。

- mysql.status[var] #这种格式可以通过传递参数来定义 Key
- mysql.ping
- mysql.version

« Template list **Template:** Template App MySQL Applications (1) Items (14) Triggers (1) Graphs (2) Screens (1)

Wizard	Name ↑	Triggers	Key
	MySQL begin operations per second		mysql.status[Com_begin]
	MySQL bytes received per second		mysql.status[Bytes_received]
	MySQL bytes sent per second		mysql.status[Bytes_sent]
	MySQL commit operations per second		mysql.status[Com_commit]
	MySQL delete operations per second		mysql.status[Com_delete]
	MySQL insert operations per second		mysql.status[Com_insert]
	MySQL queries per second		mysql.status[Questions]
	MySQL rollback operations per second		mysql.status[Com_rollback]
	MySQL select operations per second		mysql.status[Com_select]
	MySQL slow queries		mysql.status[Slow_queries]
	MySQL status	Triggers (1)	mysql.ping
	MySQL update operations per second		mysql.status[Com_update]
	MySQL uptime		mysql.status[Uptime]
	MySQL version		mysql.version

图 11-9

然后编写一个 Key 的脚本。

MySQL 监控的原理很简单, 用 MySQL 的命令 show status 即可查看相关的性能参数, 然后依次取值。

① 在 Zabbix-Agent 端添加 MySQL 的监控脚本

```
shell# vim /etc/zabbix/scripts/monitor_mysql

#!/bin/bash
#author:itnihao
#mail:itnihao@qq.com
#date 2013-12-18
#version v1.0
#function:use zabbix monitor mysql status

mysql=$(which mysql)
#注意, 如果MySQL是非标准安装, 请写出MySQL的绝对路径
#mysql=/usr/bin/mysql
var=$1
MYSQL_USER=$2      #这里的参数方便后面在Web界面中自定义变量
MYSQL_PASSWORD=$3
MYSQL_Host=$4

#如果没有传递参数, 则使用默认的用户名、密码和主机名
[ "${MYSQL_USER}" = '' ] && MYSQL_USER=zabbix
[ "${MYSQL_PASSWORD}" = '' ] && MYSQL_PASSWORD=zabbix
[ "${MYSQL_Host}" = '' ] && MYSQL_Host=localhost

[ "${var}" = '' ] && echo ""||${mysql} -u${MYSQL_USER} -p${MYSQL_P
ASSWORD} -h${MYSQL_Host} -e 'show status'|grep -v Variable_name|grep
"\b${var}\b"|awk '{print $2}'
```

测试脚本, 结果如图 11-10 所示。


```
[root@localhost zabbix_agentd.conf.d]# /etc/zabbix/scripts/monitor_mysql
[root@localhost zabbix_agentd.conf.d]# /etc/zabbix/scripts/monitor_mysql uptime
117953
[root@localhost zabbix_agentd.conf.d]#
```

图 11-10

从图 11-10 中可以看到，脚本能正常运行（如果不能正常运行，则需要检查授权信息是否正确）。

② 修改 zabbix_agentd.conf 配置文件。

```
shell# egrep -v "(^#|^$)" /etc/zabbix/zabbix_agentd.conf
#主要修改以下参数
Include=/etc/zabbix/zabbix_agentd.conf.d/      #配置文件路径
UnsafeUserParameters=1                        #允许特殊字符
```

③ 子配置文件定义 Key 的名称。

```
shell#vim/etc/zabbix/zabbix_agentd.conf.d/mysql_status.conf
UserParameter=mysql.status[*],/etc/zabbix/scripts/monitor_mysql $
1 $2 $3 $4
UserParameter=mysql.ping,/usr/bin/mysqladmin -uzabbix -pzabbix pi
ng|grep alive|wc -l
UserParameter=mysql.version,mysql -V | cut -f6 -d" " | sed 's/,/'
```

④ 对 MySQL 服务的主机添加 MySQL 模板，如图 11-11 所示。

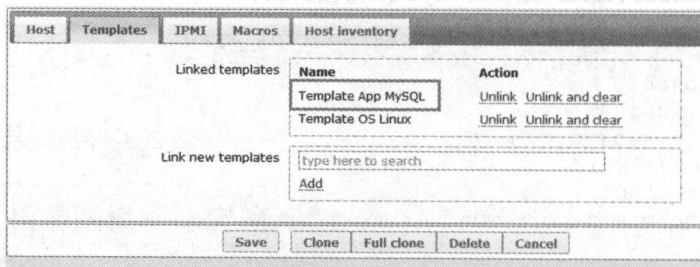


图 11-11

大约 1 分钟（即监控项的更新周期）后，就可以看到相关的监控参数，如图 11-12 所示。

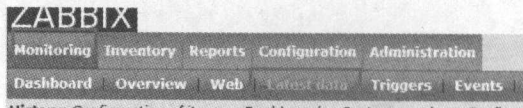


图 11-12

在 MySQL 的分组中可以看到取到的数值，如图 11-13 所示。

以上步骤完成后，就可以让默认的 MySQL 模板生效了。但是我们不会满足于此，能否添加更多的监控呢？答案是肯定的。

添加 Key，单击 Items，如图 11-14 所示。

MySQL (14 items)		
MySQL begin operations per second	17 Dec 2013 19:08:11	0 qps
MySQL bytes received per second	17 Dec 2013 19:08:09	0 Bps
MySQL bytes sent per second	17 Dec 2013 19:08:10	0 Bps
MySQL commit operations per second	17 Dec 2013 19:08:12	0 qps
MySQL delete operations per second	17 Dec 2013 19:08:13	0 qps
MySQL insert operations per second	17 Dec 2013 19:08:14	0 qps
MySQL queries per second	17 Dec 2013 19:08:18	0 qps
MySQL rollback operations per second	17 Dec 2013 19:08:15	0 qps
MySQL select operations per second	17 Dec 2013 19:08:16	0 qps
MySQL slow queries	17 Dec 2013 19:08:19	0
MySQL status	17 Dec 2013 19:08:08	Up (1)
MySQL update operations per second	17 Dec 2013 19:08:17	0 qps
MySQL uptime	17 Dec 2013 19:08:20	1 day, 08:57:01
MySQL version	17 Dec 2013 18:28:21	5.1.61

图 11-13

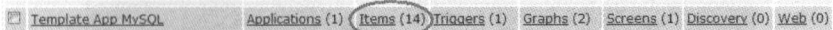


图 11-14

模板中自带的 Items 有限, 如果需要创建新的 Items, 例如, 监控总的连接次数 (Connections), 那么添加创建新的 Items 即可。对于创建新的 Items, 这里可以使用 Clone (克隆) 的小技巧, 由于数据填充内容都相似, 那么可以用 Zabbix 监控项中的 Clone (克隆) 功能, 任意单击 MySQL 模板中的一个 Items, 如图 11-15 所示。

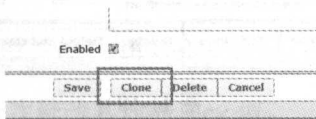


图 11-15

然后单击图 11-15 中的 “Clone” 按钮, 修改对应的填充参数即可, 如图 11-16 所示。

Item configuration details:

- Name: MySQL Connections
- Type: Zabbix agent
- Key: mysql.status[Connections]
- Type of information: Numeric (float)
- Units:
- Use custom multiplier:
- Update interval (in sec): 60
- Flexible intervals: No flexible intervals defined.
- New flexible interval: Interval (in sec) 50, Period 1-7,00:00-24:00
- Keep history (in days): 7
- Keep trends (in days): 365
- Store value: Delta (speed per second)
- Show value: As is
- New application:
- Applications: MySQL

图 11-16

对于更多的监控项参数，读者可以根据自己的需要添加，下面对 MySQL 的模板进行一些改进，在 MySQL 模板中使用宏的设置。

在有些场合，添加监控的人员对机器并没有操作的权限（不能使用 SSH 登录服务器），尤其是像数据库这么重要的服务器，一般都是禁止普通用户使用 SSH 登录的，当系统管理人员把上面的监控脚本添加到操作系统后，其他人员不一定就知道脚本中的默认密码，这时让其他人员登录 Zabbix 修改参数的配置（即用户名、密码、主机的自定义参数）就非常有意义了。

选择 MySQL 模板，对其复制（Clone）一份模板，如图 11-17 和图 11-18 所示。

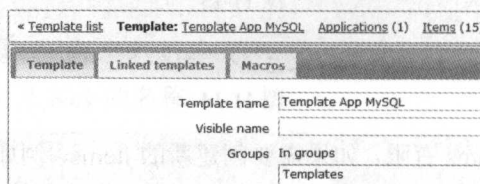


图 11-17

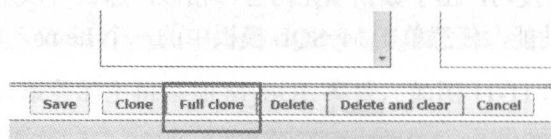


图 11-18

注意，Clone 只复制名称，不会复制模板中的 Key。

修改 Template name，如图 11-19 所示。

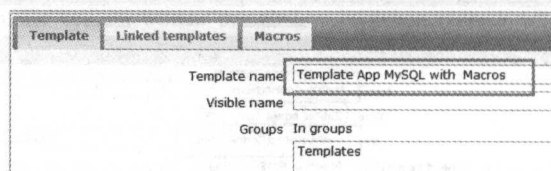


图 11-19

切换到 Macros 自定义宏变量，如图 11-20 所示。

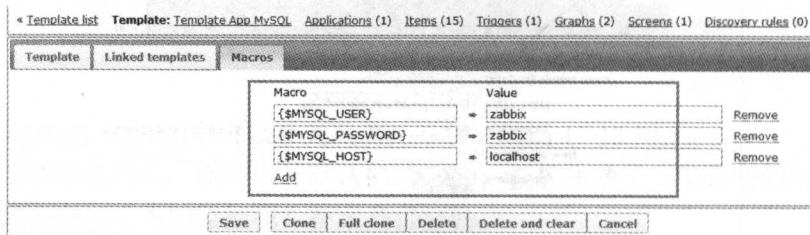


图 11-20

单击 Key，如图 11-21 所示。

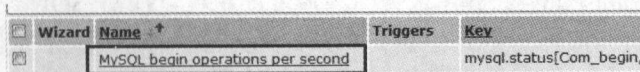


图 11-21

修改 Key，如图 11-22 所示。

```
mysql.status[Com_begin,{$MYSQL_USER},{$MYSQL_PASSWORD},{$MYSQL_HOST}]
```

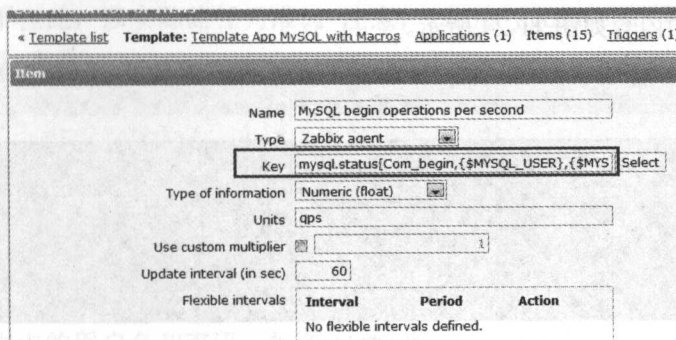


图 11-22

修改完后保存，然后依次修改其他的 Key 值。

修改完毕的 Key 值如图 11-23 所示，链接到对应的主机，如果用户名和密码发生了改变，只需要改变宏的设置即可，无须登录服务器修改脚本中的密码。

Wizard	Name	Triggers	Key	Interval
	MySQL begin operations per second		mysql.status[Com_begin,{\$MYSQL_USER},{\$MYSQL_PASSWORD},{\$MYSQL_HOST}]	60
	MySQL bytes received per second		mysql.status[Bytes_received,{\$MYSQL_USER},{\$MYSQL_PASSWORD},{\$MYSQL_HOST}]	60
	MySQL bytes sent per second		mysql.status[Bytes_sent,{\$MYSQL_USER},{\$MYSQL_PASSWORD},{\$MYSQL_HOST}]	60
	MySQL commit operations per second		mysql.status[Com_commit,{\$MYSQL_USER},{\$MYSQL_PASSWORD},{\$MYSQL_HOST}]	60
	MySQL Connections		mysql.status[Connections,{\$MYSQL_USER},{\$MYSQL_PASSWORD},{\$MYSQL_HOST}]	60
	MySQL delete operations per second		mysql.status[Com_delete,{\$MYSQL_USER},{\$MYSQL_PASSWORD},{\$MYSQL_HOST}]	60
	MySQL insert operations per second		mysql.status[Com_insert,{\$MYSQL_USER},{\$MYSQL_PASSWORD},{\$MYSQL_HOST}]	60
	MySQL queries per second		mysql.status[Questions,{\$MYSQL_USER},{\$MYSQL_PASSWORD},{\$MYSQL_HOST}]	60
	MySQL rollback operations per second		mysql.status[Com_rollback,{\$MYSQL_USER},{\$MYSQL_PASSWORD},{\$MYSQL_HOST}]	60
	MySQL select operations per second		mysql.status[Com_select,{\$MYSQL_USER},{\$MYSQL_PASSWORD},{\$MYSQL_HOST}]	60
	MySQL slow queries		mysql.status[Slow_queries,{\$MYSQL_USER},{\$MYSQL_PASSWORD},{\$MYSQL_HOST}]	60

图 11-23

11.4.2 用 Percona Monitoring Plugins 监控 MySQL

Percona 对 MySQL 数据库服务器进行了改进，在功能和性能上较 MySQL 有显著的提升。该版本提升了在高负载情况下 InnoDB 的性能，为 DBA 提供了一些

非常有用的性能诊断工具。另外，有更多的参数和命令来控制服务器行为。

Percona Monitoring Plugins 是一个高质量的组件，为 MySQL 数据库添加企业级的监控和图表功能。另外，该插件可以和 Nagios 或者是 Cacti 等监控系统集成，从 Percona Monitoring Plugins 1.1 开始，支持 Zabbix 的监控。其脚本使用 PHP 实现，故需要安装 PHP 环境。

下载 Percona Monitoring Plugins。

```
shell# wget http://www.percona.com/redir/downloads/percona-monitoring-plugins/LATEST/percona-zabbix-templates-1.1.1-1.noarch.rpm
```

解压后的内容如图 11-24 所示。

```
shell# rpm2cpio percona-zabbix-templates-1.1.1-1.noarch.rpm | cpio -div
```

```
[root@localhost percona-zabbix-templates]# ll
total 32
-rw-r--r-- 1 root root 30432 Dec 30 19:30 percona-zabbix-templates-1.1.1-1.noarch.rpm
[root@localhost percona-zabbix-templates]# rpm2cpio percona-zabbix-templates-1.1.1-1.noarch.rpm | cpio -div
./var/lib/zabbix/percona
./var/lib/zabbix/percona/scripts
./var/lib/zabbix/percona/scripts/get_mysql_stats_wrapper.sh
./var/lib/zabbix/percona/scripts/ss_get_mysql_stats.php
./var/lib/zabbix/percona/templates
./var/lib/zabbix/percona/templates/userparameter_percona_mysql.conf
./var/lib/zabbix/percona/templates/zabbix_agent_template_percona_mysql_server_ht_2.0.9-sver1.1.1.xml
681 blocks
```

图 11-24

目录结构说明如下。

```
#脚本文件路径
/var/lib/zabbix/percona/scripts/get_mysql_stats_wrapper.sh
/var/lib/zabbix/percona/scripts/ss_get_mysql_stats.php
#key文件位置
/var/lib/zabbix/percona/templates/userparameter_percona_mysql.conf
#模板文件位置
/var/lib/zabbix/percona/templates/zabbix_agent_template_percona_mysql_server_ht_2.0.9-sver1.1.1.xml
```

安装 percona-zabbix-templates，如图 11-25 所示。

```
shell# rpm -ivh percona-zabbix-templates-1.1.1-1.noarch.rpm
```

```
[root@localhost percona-zabbix-templates]# rpm -ivh percona-zabbix-templates-1.1.1-1.noarch.rpm
warning: percona-zabbix-templates-1.1.1-1.noarch.rpm: Header V4 DSA/SHA1 Signature, key ID cd2efd2a: NOKEY
Preparing... [100%]
1:percona-zabbix-templates [100%]
Scripts are installed to /var/lib/zabbix/percona/scripts
Templates are installed to /var/lib/zabbix/percona/templates
[root@localhost percona-zabbix-templates]#
```

图 11-25

如果没有安装 PHP，需要安装 php 和 php-mysql 软件包，如图 11-26 所示。


```
[root@localhost percona-zabbix-templates]# rpm -qa |grep php
php-pdo-5.3.3-3.el6_2.8.x86_64
php-gd-5.3.3-3.el6_2.8.x86_64
php-5.3.3-3.el6_2.8.x86_64
php-xmlrpc-5.3.3-3.el6_2.8.x86_64
php-cli-5.3.3-3.el6_2.8.x86_64
php-bcmath-5.3.3-3.el6_2.8.x86_64
php-mysql-5.3.3-3.el6_2.8.x86_64
php-common-5.3.3-3.el6_2.8.x86_64
php-mbstring-5.3.3-3.el6_2.8.x86_64
php-xml-5.3.3-3.el6_2.8.x86_64
```

图 11-26

将定义 Key 的子配置文件复制到/etc/zabbix/zabbix_agentd.conf.d/下（如果你的安装路径不在此，以实际路径为准）。注意，zabbix_agentd.conf 配置文件中需要开启 include 参数。

```
shell# cp /var/lib/zabbix/percona/templates/userparameter_percona
mysql.conf /etc/zabbix/zabbix_agentd.conf.d/
```

重启服务 zabbix_agentd。

```
shell# service zabbix-agent restart
Shutting Zabbix agent: [ OK ]
Starting Zabbix agent: [ OK ]
```

修改脚本中的用户名和密码，如图 11-27 所示。

```
#vim /var/lib/zabbix/percona/scripts/ss_get_mysql_stats.php
```

```
# =====
$mysql_user = 'cactiuser';
$mysql_pass = 'cactiuser';
$mysql_port = 3306;
$mysql_ssl = FALSE; # whether to use SSL to connect to MySQL.
$mysql_ssl_key = '/etc/pki/tls/certs/mysql/client-key.pem';
$mysql_ssl_cert = '/etc/pki/tls/certs/mysql/client-cert.pem';
$mysql_ssl_ca = '/etc/pki/tls/certs/mysql/ca-cert.pem';
```

图 11-27

修改为 MySQL 中存在的用户，这里的用户名和密码均为 zabbix，如果没有用户，请添加，如图 11-28 所示。

```
mysql> grant process,super,select on *.* to zabbix@localhost id
entified by 'zabbix';
mysql> flush privileges;
```

```
# =====
$mysql_user = 'zabbix';
$mysql_pass = 'zabbix';
$mysql_port = 3306;
$mysql_ssl = FALSE; # whether to use SSL to connect to MySQL.
$mysql_ssl_key = '/etc/pki/tls/certs/mysql/client-key.pem';
$mysql_ssl_cert = '/etc/pki/tls/certs/mysql/client-cert.pem';
$mysql_ssl_ca = '/etc/pki/tls/certs/mysql/ca-cert.pem';
```

图 11-28

对脚本进行调试。


```
shell# /var/lib/zabbix/percona/scripts/get_mysql_stats_wrapper.sh
gg
17306
```

以上为正常状态，说明密码配置正确，异常状态如下。

```
shell# /var/lib/zabbix/percona/scripts/get_mysql_stats_wrapper.sh
gg
ERROR: run the command manually to investigate the problem: /usr/bin/php -q /var/lib/zabbix/percona/scripts/ss_get_mysql_stats.php --host localhost --items gg
shell# /usr/bin/php -q /var/lib/zabbix/percona/scripts/ss_get_mysql_stats.php --host localhost --items gg
ERROR: Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock'
```

测试能否获取到值。

```
shell# zabbix_get -s 127.0.0.1 -k MySQL.binary-log-space
0
```

可以看到，在本机可以获取到数据。Key 的名称来源如图 11-29 所示。

```
[root@localhost percona-zabbix-templates]# cat /etc/zabbix/zabbix_agentd.conf.d/userparameter_percona_mysql.conf
UserParameter=mysql.scan,/var/lib/zabbix/percona/scripts/get_mysql_stats_wrapper.sh kt
UserParameter=mysql.slave-stopped,/var/lib/zabbix/percona/scripts/get_mysql_stats_wrapper.sh jh
UserParameter=mysql.com-replace,/var/lib/zabbix/percona/scripts/get_mysql_stats_wrapper.sh jz
UserParameter=mysql.innodb-lock-structs,/var/lib/zabbix/percona/scripts/get_mysql_stats_wrapper.sh lp
```

图 11-29

导入模板，如图 11-30 到图 11-34 所示。

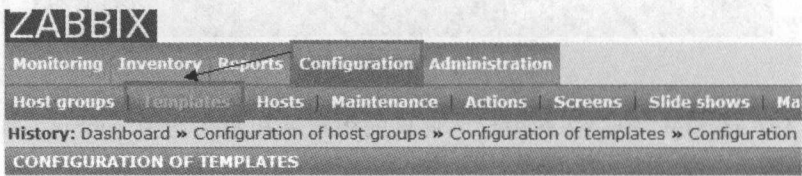


图 11-30



图 11-31

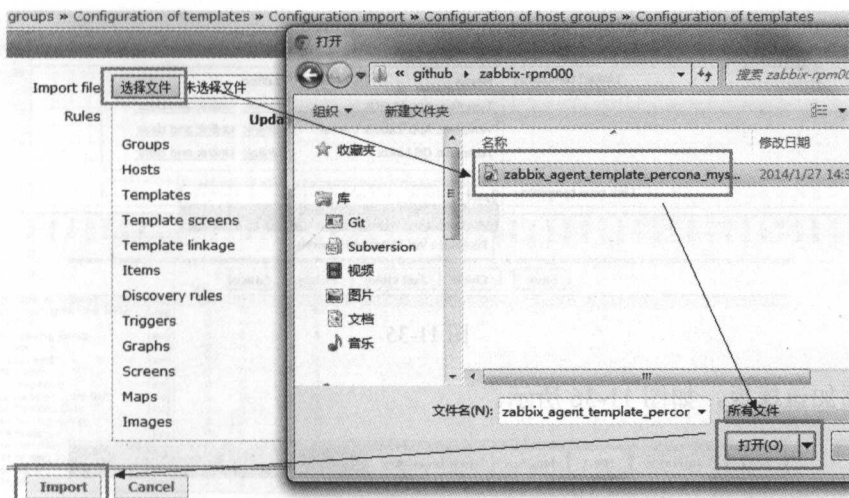


图 11-32

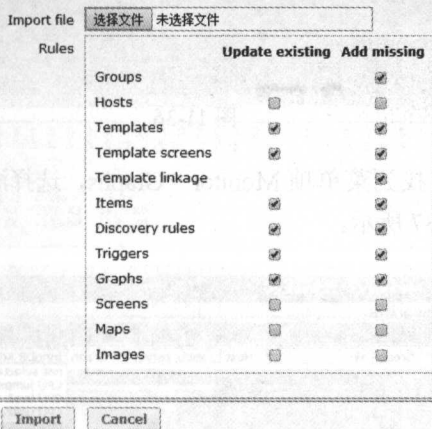
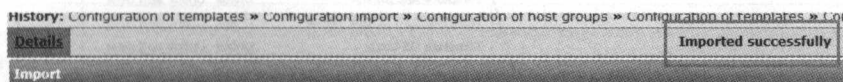


图 11-33

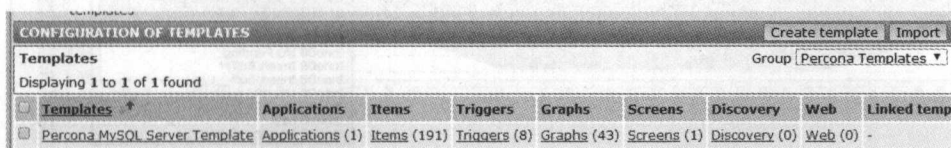


图 11-34

导入完成后，将该模板应用于主机，如图 11-35 所示。

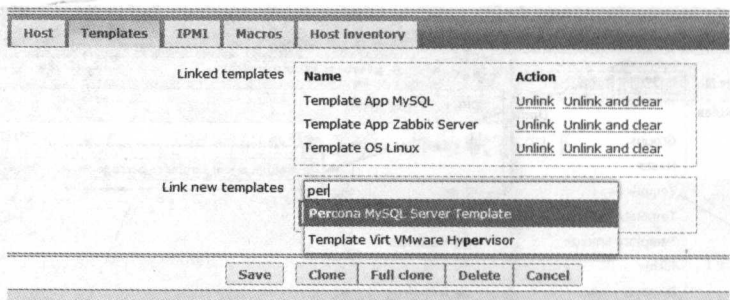


图 11-35

添加该模板，如图 11-36 所示。

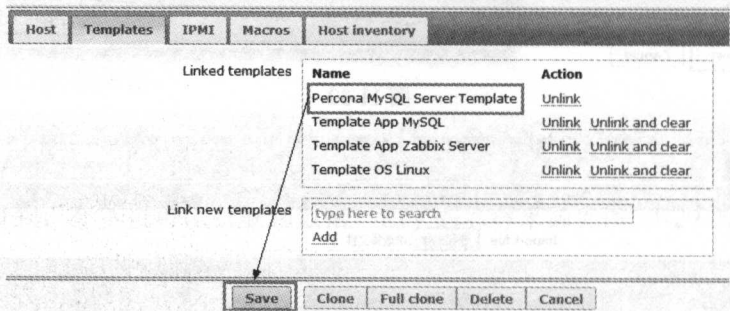


图 11-36

打开 Web 界面，找到菜单项 Monitor→Graphs，选择添加 MySQL 模板的主机，可以看到图形如图 11-37 所示。

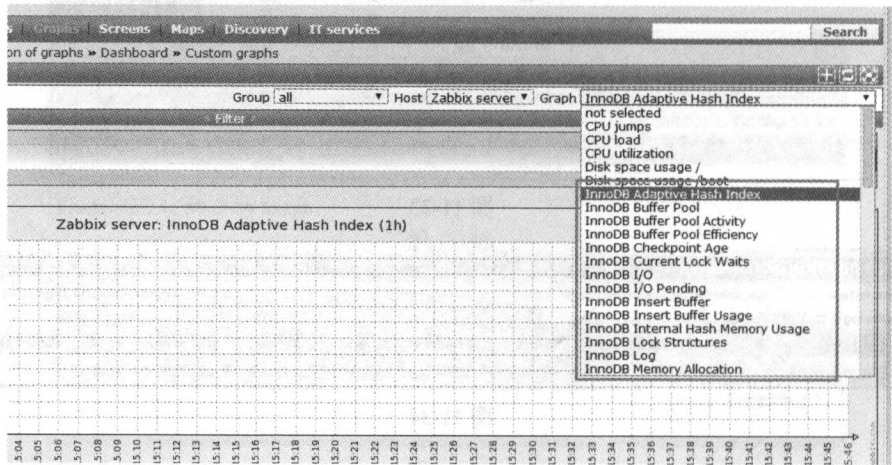


图 11-37

监控数据如图 11-38 至图 11-40 所示。

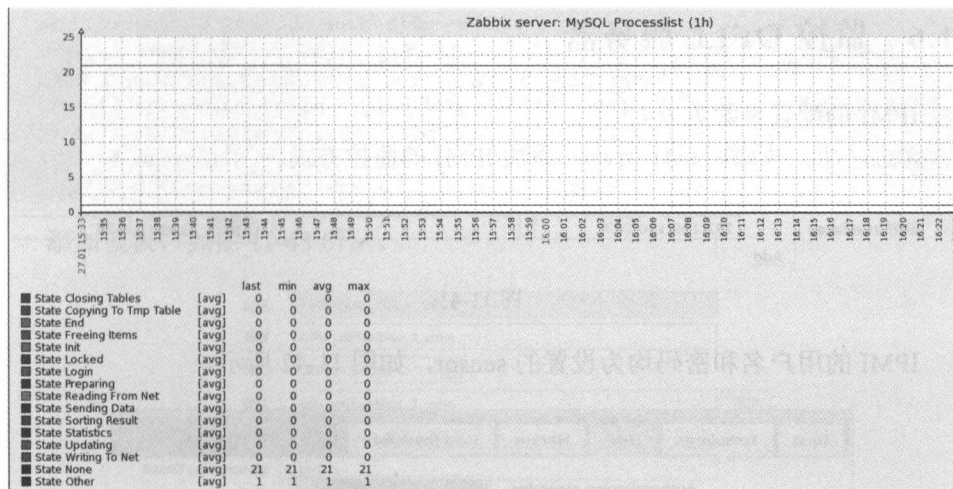


图 11-38

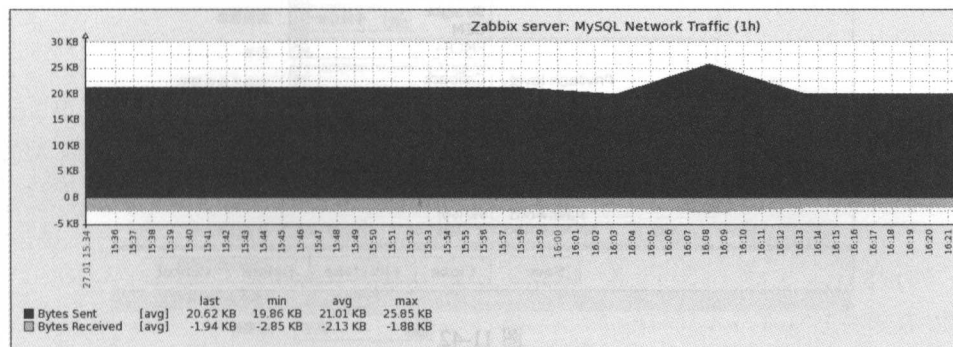


图 11-39

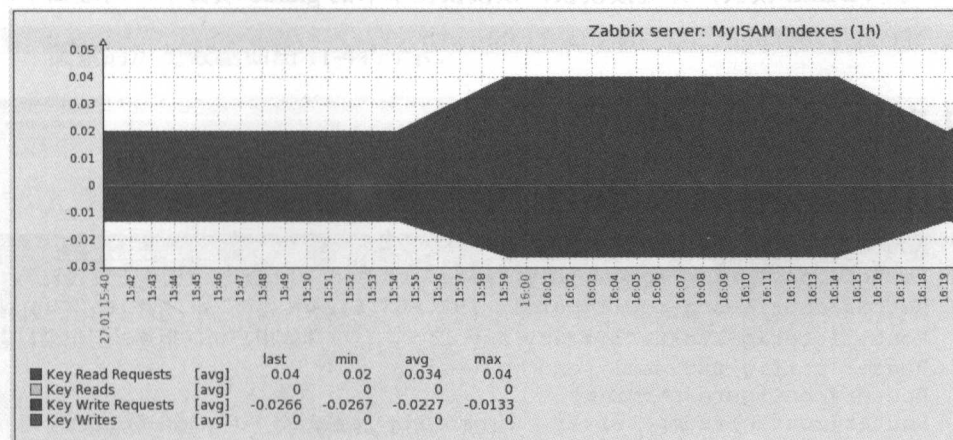


图 11-40

11.6 监控 DELL 服务器

IPMI 的配置参数见 6.4.3 节。

添加主机，如图 11-41 所示，添加 IPMI 的监控方式。

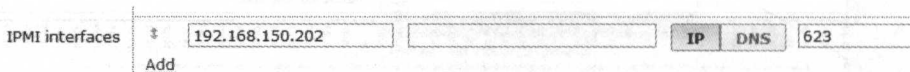


图 11-41

IPMI 的用户名和密码均为设置的 sensor，如图 11-42 所示。

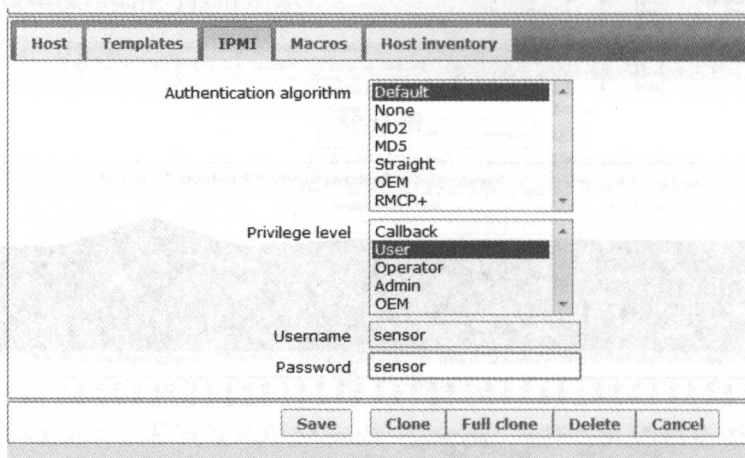


图 11-42

导入 DELL 模板，并链接模板，该模板在本书的 github 项目中，网址如下。

<https://github.com/itnihao/zabbix-book/tree/master/11-chapter>

11.7 监控 Cisco 路由器

配置 Cisco 的 SNMP 如下（其他更多的设置请参考官方文档）。

```
Router#configure terminal
Router(config)#ip access-list standard snmp-filter #创建方位列表
Router(config-std-nacl)#permit 192.168.0.240#允许192.168.0.240访问
Router(config-std-nacl)#deny any log #其他用户访问拒绝，但记录日志
Router(config-std-nacl)#end
Router#configure terminal
Router(config)#snmp-server community public RO snmp-filter
#设置团组名 (community) 为public
```

用 snmpwalk 测试 SNMP 的配置语句如下。


```
shell# yum install net-snmp-utils -y
shell# snmpwalk -v 2c -c public 172.30.31.10 1.3.6.1.4.1.9.2.1.57.0
SNMPv2-SMI::enterprises.9.2.1.57.0 = INTEGER: 8
shell# snmpwalk -v 2c -c public 172.30.31.10 1.3.6.1.4.1.9.2.1.57.0
SNMPv2-SMI::enterprises.9.2.1.57.0 = INTEGER: 8
shell# snmpwalk -v 2c -c public 172.30.31.10 1.3.6.1.4.1.9.2.1.57.0
SNMPv2-SMI::enterprises.9.2.1.57.0 = INTEGER: 8
```

添加模板，如图 11-43 所示。

主机	Template_Cisco_3560								
名称	CPU utilization 1 min								
类型	SNMPv2 代理								
键值	cpu_utilization_1_min		选择						
SNMP OID	1.3.6.1.4.1.9.2.1.57.0								
SNMP community	pubic								
端口									
数据类型	数字的 (无正负)								
数据类型	十进制数字								
单位	%								
使用自定义倍数	<input type="checkbox"/> 1								
数据更新间隔(秒)	30								
弹性区间	<table><thead><tr><th>间隔</th><th>期间</th><th>动作</th></tr></thead><tbody><tr><td colspan="3">并无定义的弹性区间</td></tr></tbody></table>			间隔	期间	动作	并无定义的弹性区间		
间隔	期间	动作							
并无定义的弹性区间									
新的弹性区间	间隔(秒计)	50	期间 1-7,00:00-24:00 添加						
保留历史(日计)	7								
保留趋势(日计)	365								
保存值	不变								
显示值	不变 显示值对应								

图 11-43

流量的历史数据如图 11-44 所示。

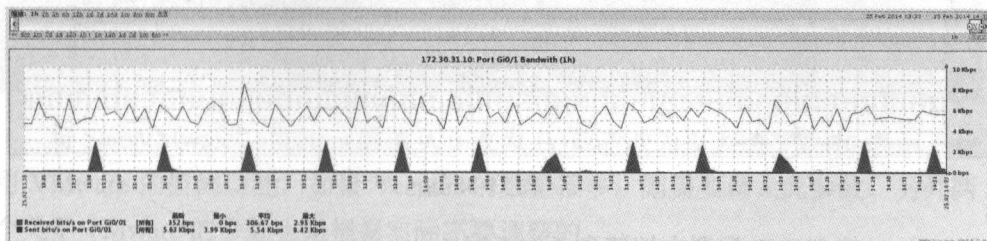


图 11-44

风扇状态的历史数据如图 11-45 所示。

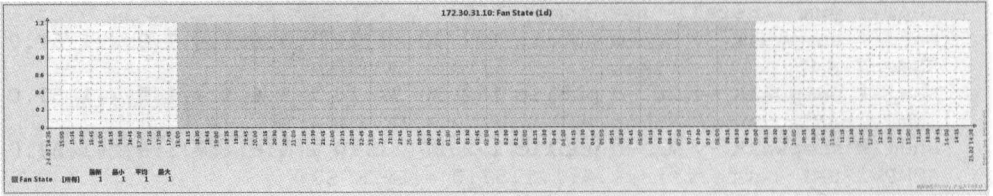


图 11-45

温度变化的历史监控数据如图 11-46 所示。

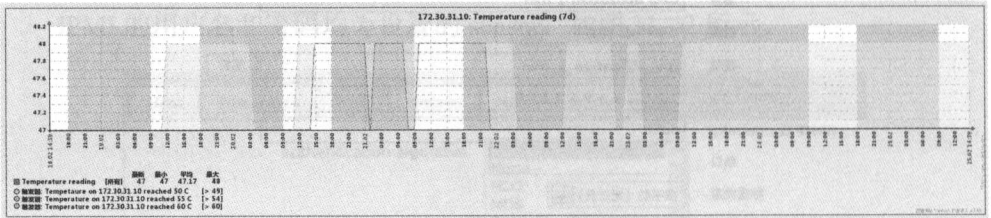


图 11-46

CPU 使用率的历史数据如图 11-47 所示。

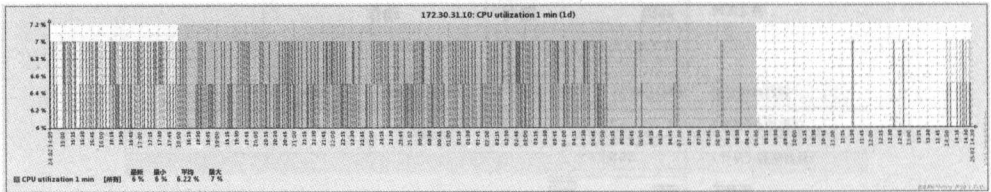


图 11-47

内存使用率的历史数据如图 11-48 所示。

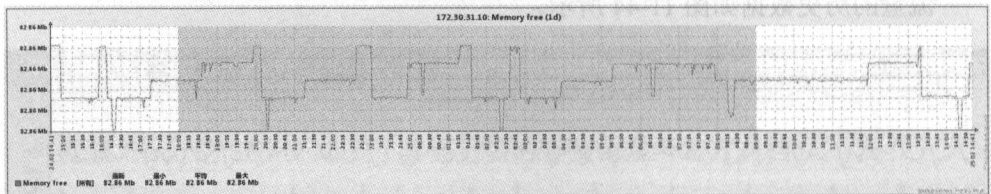


图 11-48

Cisco 的 SNMP 配置文档请参考如下网址：

http://www.cisco.com/c/en/us/td/docs/ios/12_2/configfun/configuration/guide/ffun_c/fcf014.html

11.8 监控 VMware

从 Zabbix 2.2 起, Zabbix 开始支持 VMware 产品的监控, 而且是全自动化的, 基于 low level discovery。

安装配置如下。
Zabbix-Server 在使用源码 configure 编译参数中必须加上--with-libxml2 和 --with-libcurl (官方提供的 RPM 安装包默认开启了这两个参数), 否则不会支持 VMware 的监控, 语句如下:

```
shell# /etc/init.d/zabbix_server restart
Shutting down zabbix_server: [ OK ]
Starting zabbix_server: zabbix_server [19571]: ERROR: cannot start vmware collector because Zabbix server is built without VMware support [FAILED]
```

服务端的配置文件修改如表 11-1 所示。

表 11-1

选 项	值		描 述
	范围	默认值	
StartVMwareCollectors	0~250	0	预启动的 VMware 数据采集线程数量
VMwareCacheSize	256K~2G	8M	VMware 的数据监测缓存大小 zabbix[vmware,buffer,...]可以用来查询使用内容的大小
VMwareFrequency	10~86400	60	数据采集的频率

```
shell# vim /etc/zabbix/zabbix_server.conf
### Option: StartVMwareCollectors
# Number of pre-forked vmware collector instances.
#
# Mandatory: no
# Range: 0-250
# Default:
# StartVMwareCollectors=0
StartVMwareCollectors=1
```

将 StartVMwareCollectors 的值从 0 改为 1, 或者改为更大的参数, 0 表示不启用 VMware 的监测功能, 如果 Vcenter 的数量很多, 则需要调大此参数。另外两个关于 VMware 的参数也是根据实际需要调整的。

登录 Vcenter, 确认用户名和密码, 以及服务是否正常, 如图 11-49 所示。
登录 Vcenter 后, 发现主机如图 11-50 所示。可以看到, Vcenter 和 VMware 都是正常工作的。下面开始在 Zabbix 中配置 VMware 的监控。

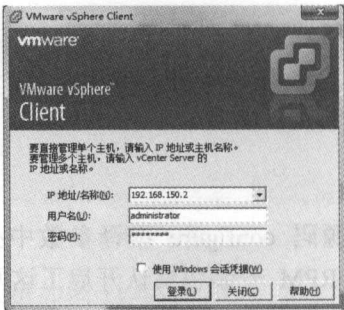


图 11-49

添加主机，如图 11-51 所示。



图 11-50

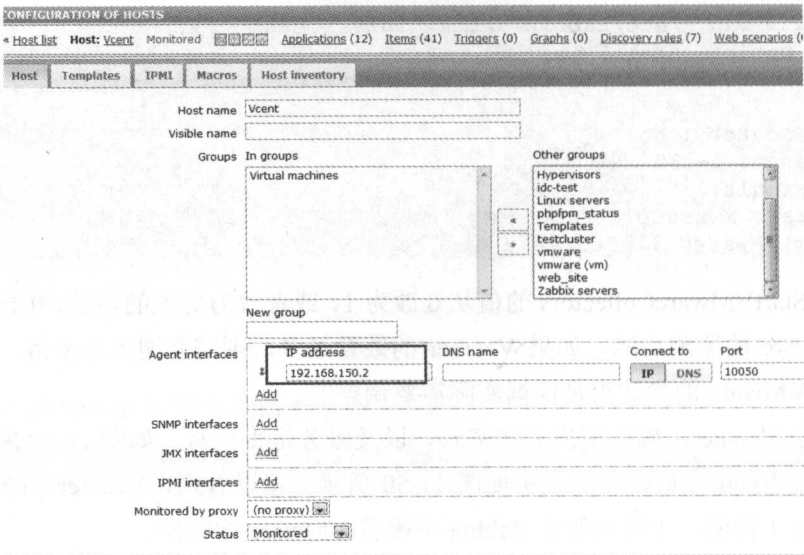


图 11-51

图 11-51 中的“IP address”可以随便填写，无须填写真实的主机 IP，但如果不填写 IP 地址，系统会报错，默认是 127.0.0.1，也可以不修改。这里修改的目的是便于以后识别监控的主机。

选择 VMware 的模板，如图 11-52 所示。

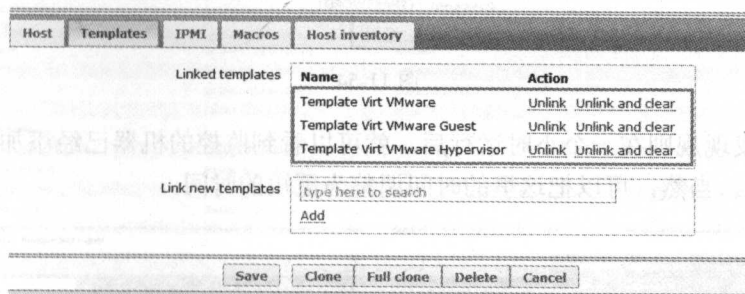


图 11-52

在图 11-52 中，模板名称及作用如表 11-2 所示。

表 11-2

模 板 名	作 用
Template Virt VMware	监控 Vcenter
Template Virt VMware Guest	监控虚拟机
Template Virt VMware Hypervisor	监控物理机

设置变量如图 11-53 所示。

{ \$URL } - VMware (vCenter 或者 vSphere) SDK 的 URL (https://X.X.X.X/sdk)
 { \$USERNAME } - VMware 的用户名
 { \$PASSWORD } - VMware 的用户密码

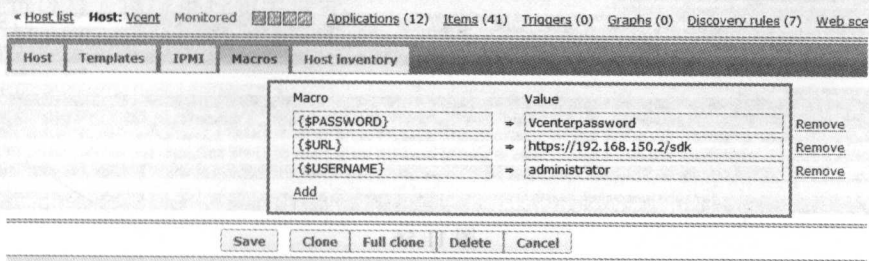


图 11-53

单击 Save 保存设置。

默认的自动发现为一个小时（即 3600s），如图 11-54 所示。

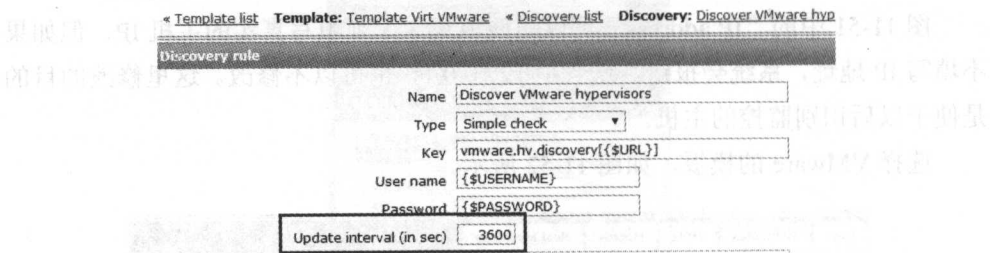


图 11-54

自动发现规则在一个小时运行后，就可以看到监控的机器已经添加了，如图 11-55 所示。当然，可以把这里的时间调整为更短的时间。

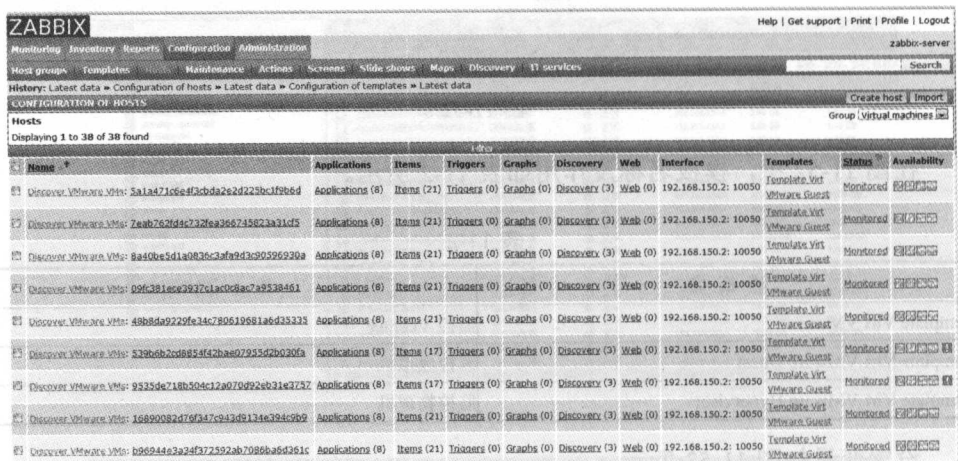


图 11-55

自动发现的 Hypervisors 如图 11-56 所示。

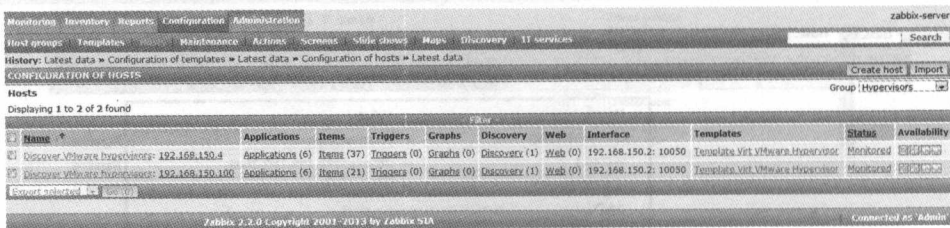


图 11-56

在 Latest data 中，可以看到获取的当前数据，如图 11-57 所示。

ZABBIX					Help Get support
Monitoring Inventory Reports Configuration Administration					
Dashboard Overview Web Latest data Triggers Events Graphs Screens Maps Discovery IT services					
History: Latest data » Configuration of hosts » Latest data » Configuration of hosts » Latest data					
LATEST DATA					
Items					Group: vmware (vm) Host: j-2-S053-YM
Name	Last check	Last value	Change		
CPU (2 items)					
CPU usage	04 Dec 2013 17:30:12	0 Hz	-		Graph
Number of virtual CPUs	04 Dec 2013 17:30:11	2	-		Graph
General (4 items)					
Cluster name	04 Dec 2013 16:40:10	vmware	-		History
Hypervisor name	04 Dec 2013 16:40:13	192.168.150.4	-		History
Power state	04 Dec 2013 17:30:27	poweredOff (0)	-		Graph
Uptime	04 Dec 2013 17:30:31	00:00:00	-		Graph
Memory (8 items)					
Ballooned memory	04 Dec 2013 17:30:14	0 B	-		Graph
Compressed memory	04 Dec 2013 17:30:15	0 B	-		Graph
Guest memory usage	04 Dec 2013 17:30:19	0 B	-		Graph
Host memory usage	04 Dec 2013 17:30:20	0 B	-		Graph
Memory size	04 Dec 2013 17:30:22	4 GB	-		Graph

图 11-57

查看图形，如图 11-58 所示。

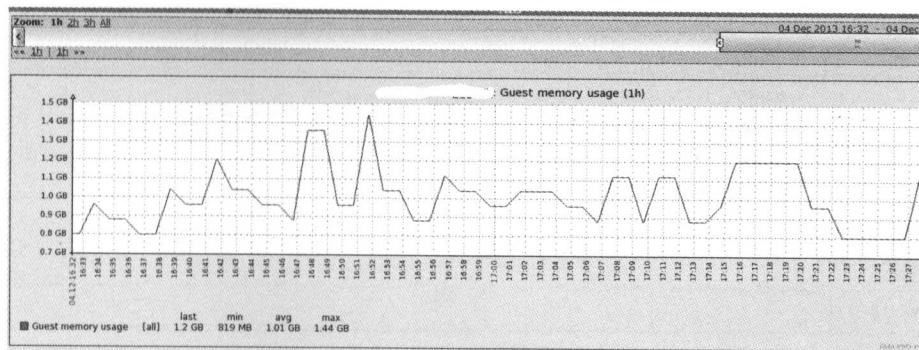


图 11-58

有关Hadoop监控的内容，请参考项目地址<https://code.google.com/p/mikoomi/>，读者可以导入模板进行相关设置。

更多的监控模板和方法请读者参考以下链接。

<https://github.com/itnihao/zabbix-book/tree/master/11-chapter>
https://www.zabbix.org/wiki/Zabbix_Templates

第3部分

高级部分

第 12 章 性能优化

在 Zabbix 的使用过程中，随着监控对象的增多，Zabbix-Server 会面临非常大的压力，从而让 Zabbix 的管理人员面临管理的难题。因此，本章从性能优化方面给出了一些具体解决方案。

12.1 Zabbix 性能优化概述

造成 Zabbix 性能下降的因素如表 12-1 所示。

表 12-1

因 素	慢	快
数据库大小	巨大	适应内存大小
触发器表达式的复杂程度	Min()、max()、avg()	Last()、nodata()
数据收集方法	轮询（SNMP、无代理、Passive 代理）	Trapping（Active 代理）
数据类型	文本、字符串	数值
前端用户数量	多	少

主机数量也是影响性能的主要因素，如表 12-2 所示。

表 12-2

	主机数量（台）	性能（NVPS）
每个主机有 60 个 Items 每分钟更新一次	10	10
	100	100
	1000	1000
每个主机有 600 个 Items 每分钟更新一次	10	100
	100	1000
	1000	10000

Zabbix 性能低下的表现如下。

- Zabbix队列中有太多被延迟的Item，通过菜单项查看：Administration→Queue 查看。

- Zabbix绘图中经常出现断图，一些Item没有数据。
- 带有nodata()函数的触发器出现False。
- 前端页面无响应，或者响应很慢。

解决方案如下。

- 不要使用默认的模板，应定制自己的模板。
- 数据库调优。
- 架构优化，如使用分布式，各服务器功能独立。
- Items、Trigger调优。
- 更换更好的硬件。

12.2 Zabbix 性能优化的依据

对 Zabbix-Sever 本身进行监控，选择 Zabbix-Server 的监控模板，如图 12-1 所示。

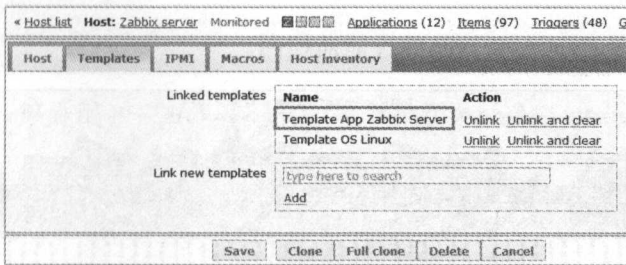


图 12-1

看到如图 12-2 所示的 Zabbix-Server 内部监控情况。

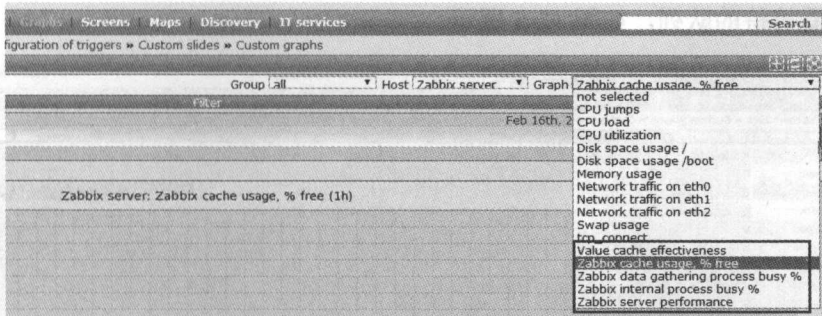


图 12-2

关于 Cache 的性能监控，如图 12-3 所示。

监控的指标是剩余的容量，如果剩余容量已非常小，可以通过调大 zabbix_server.conf 中的缓存参数，直到这里的监控数据有剩余量为止。

Zabbix-Server 性能的监控有以下两个。

- 每秒处理的数量。
- 等待的队列数量。

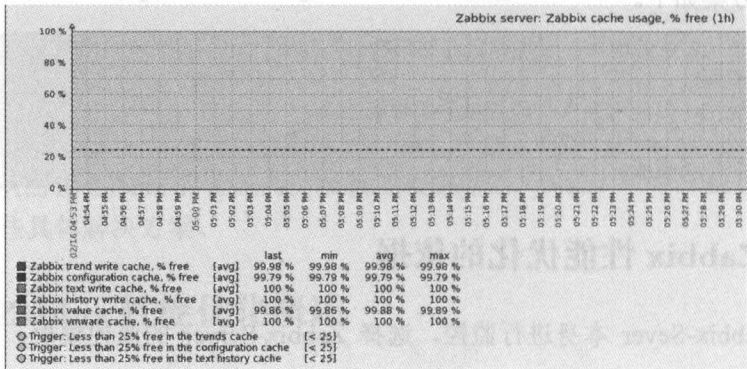


图 12-3

如图 12-4 是 Zabbix-Server 的性能监控图。

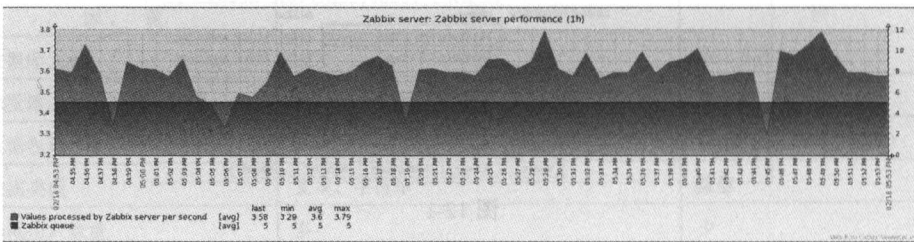


图 12-4

查看等待的队列，如图 12-5 所示。

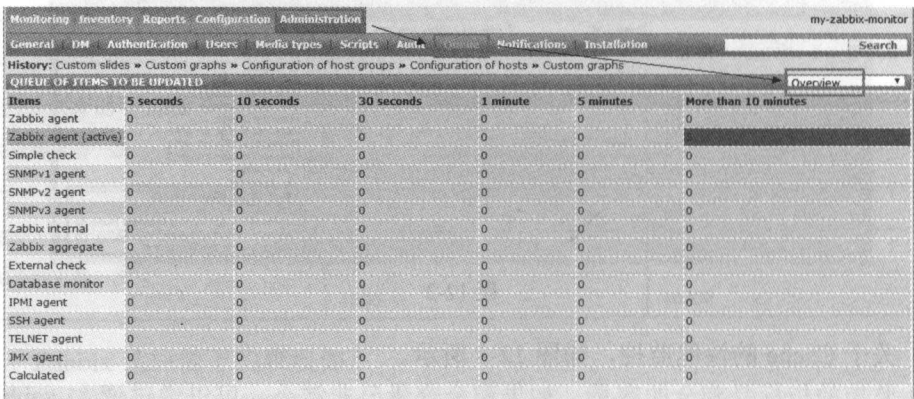


图 12-5

查看详情，可以看到具体的 Items，如图 12-6 所示。

Monitoring Inventory Reports Configuration Administration my-zabbix-monitor

General DM Authentication Users Media types Scripts Audit Queue Configurations Installation Search

History: Custom graphs » Configuration of host groups » Configuration of hosts » Custom graphs » Queue

QUEUE OF ITEMS TO BE UPDATED

Scheduled check

Delayed by

Host

Name

Feb 16th, 2014 03:17:47 PM 2h 38m 32s

devops.itnihao.com

Agent ping

Feb 16th, 2014 03:17:47 PM 2h 38m 32s

beijing-test-001

Total memory

Feb 16th, 2014 03:17:47 PM 2h 38m 32s

Zabbix server1

Total memory

Feb 16th, 2014 04:16:47 PM 1h 39m 32s

devops.itnihao.com

Version of zabbix_agent(d) running

Feb 16th, 2014 04:16:47 PM 1h 39m 32s

devops.itnihao.com

Host name of zabbix_agentd running

图 12-6

等待的队列越多，说明 Zabbix-Server 性能越差。

NVPS 的计算语句如下。

```
mysql> SELECT SUM(1.0/i.delay) AS qps FROM items i,hosts h WHERE
i.status='ITEM_STATUS_ACTIVE' AND i.hostid=h.hostid AND h.status='HOST_STATUS_MONITORED' ;
+-----+
| qps    |
+-----+
| 53.24389 |
```

NVPS 的计算是用 PHP 语言实现的，文件位于 include/func.inc.php 的第 2509 行（不同的版本，位置可能不相同），如图 12-7 所示。

```
// comments: !!! Don't forget sync code with C !!!
$row = DBfetch(DBselect(
    'SELECT SUM(1.0/i.delay) AS qps',
    'FROM items i,hosts h',
    'WHERE i.status='.ITEM_STATUS_ACTIVE,
    'AND i.hostid=h.hostid',
    'AND h.status='.HOST_STATUS_MONITORED,
    'AND i.delay<=0',
    'AND i.flags<>1.ZBX_FLAG_DISCOVERY_PROTOTYPE
));
$status['qps_total'] = round($row['qps'], 2);

return $status;
}
```

图 12-7

12.3 配置文件的参数优化

在 Zabbix-Server 中，关于配置文件的参数调整，可以配置的内容如下。

- 进程的数量。
- 缓存的大小。
- 超时时间。

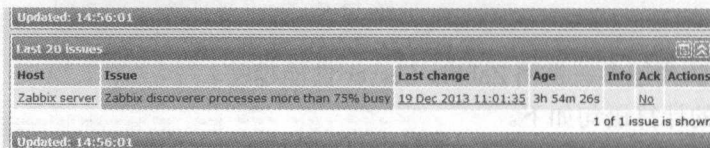
例如，对参数进行如下调整。

```
StartPollers=160
StartPollersUnreacheable=80
StartTrappers=20
StartPingers=100
StartDiscoverers=120
```

```
CacheSize=1024M
StartDBSyncers=16
HistoryCacheSize=1024M
TrendCacheSize=1024M
HistoryTextCacheSize=512M
```

对这些参数进行合理的设置会使 Zabbix 处于正常的工作状态。

例如, 开启 Discovery, 在设置数量多的情况下, 默认参数通常不能满足需求, 会触发设置的阈值, 如图 12-8 所示。



Host	Issue	Last change	Age	Info	Ack	Actions
Zabbix server	Zabbix discoverer processes more than 75% busy	19 Dec 2013 11:01:35	3h 54m 26s	No		

图 12-8

提示发现进程繁忙时, 需要修改/etc/zabbix/zabbix_server.conf, 如图 12-9 所示。

```
### Option: StartDiscoverers
# Number of pre-forked instances of discoverers.
#
# Mandatory: no
# Range: 0-250
# Default: 1
# StartDiscoverers=1
```

图 12-9

设置 StartDiscoverers=20, 或者更高, 最大值是 250, 根据实际需要增大值, 建议设置 100 左右, 值越大, 消耗的 CPU 和内存越多。

修改配置文件后, 需要重启 Zabbix-Server 的进程, 如图 12-10 所示。

```
[root@localhost ~]# /etc/init.d/zabbix_server restart
Shutting down zabbix_server:
Starting zabbix_server:
```

图 12-10

对于内存大小的设置, 其和共享内存的大小是有关系的, 参考如下网址的内容。

https://www.zabbix.org/wiki/How_to/configure_shared_memory

另一个需要注意的问题是, 在自动发现规则中, 运行时间也有关系, 在 network discovery 和 low level discovery 中, 默认的自动发现时间 Delay (in sec) 是 3600s, 如果设置得太长, 则自动发现不能及时运行发现规则, 但在实际环境中, 希望在更短的时间内运行发现规则, 能够及时添加主机, 能够发现添加硬盘和网卡的监控项, 将这个时间设置得更短, 建议在发现规则运行完毕后将此值设置为更大的值, 如 3600 等, 如图 12-11 所示, 或者直接将自动化发现功能关闭, 避免资源的浪费。

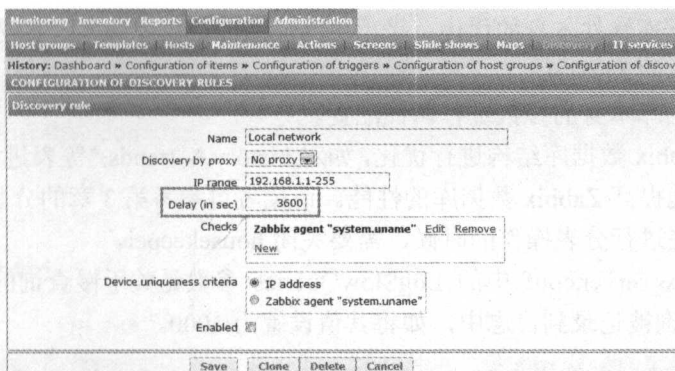


图 12-11

对于其他配置参数，请读者依此类推，根据实际需要调整即可。

12.4 Zabbix 的架构优化

Zabbix 的架构可以分为以下三种模式。

- Server/Agent 模式。
- Server/Node/Agent 模式。
- Serve/Proxy/Agent 模式。

通过采用分布式模式，可以大大降低 Server（服务器）的负担，从而大大提升单台 Server（服务器）的性能。关于 Proxy 和 Node 模式的内容，请读者参考相关章节。

12.5 Items 工作模式及 Trigger 的优化

Zabbix 中的 Items 默认工作于被动模式，可以通过设置主动模式来提高 Server 的性能。主动模式的配置见 8.3.2 节。

Trigger 中正则表达式函数 last()、nodata() 的速度是最快的，min()、manx()、avg() 是最慢的，尽量使用速度快的函数。

在 Trigger 的配置中，很可能出现由于一个函数的逻辑错误导致数据库查询较慢的现象。

12.6 Zabbix 的数据库优化

Zabbix 数据库的优化可分为以下几部分。

- 对数据库软件本身的优化。采用更高性能的数据库版本，如选择 Percona，选用最新版本的 MySQL。

- 对数据库本身的参数进行调优配置。

- 对 Zabbix 数据库结构进行优化，如对 `history.*`、`trends.*` 等表进行分表操作，会最大程度地提高 Zabbix 数据库的性能。请读者可参考第 3 章的分表内容。

注意，在进行分表操作的时候，需要关闭 `housekeeper`。

在 `zabbix_server.conf` 中的 `LogSlowQueries` 参数是关于慢查询的设置，超过 1000ms 的查询被记录到日志中，如将其值设置为 1000。

```
LogSlowQueries=1000
```

开启这个参数可以对慢查询继续记录，方便调优配置。

- 对数据库操作系统的优化。

下面给一个 MySQL 参数配置的例子，仅供读者参考使用（在 32GB 内存的服务器中）。

```
shell#_egrep -v "(#|^$)" /etc/my.cnf
[client]
port                = 3306
socket              = /var/lib/mysql/mysql.sock
[mysqld]
port                = 3306
socket              = /var/lib/mysql/mysql.sock
skip-external-locking
character-set-server = utf8
slow_query_log
slow_query_log_file = /var/log/mysql/slowquery.log
long_query_time = 2
log_error = /var/log/mysql/log-error.log
skip-name-resolve
max_connections = 5000
back_log = 300
table-cache=4096
max_allowed_packet = 32M
max-heap-table-size = 128M
key_buffer_size     = 128M
sort-buffer-size    = 16M
join-buffer-size    = 16M
net_buffer_length   = 8K
read_buffer_size    = 256K
read_rnd_buffer_size = 512K
mysam_sort_buffer_size = 8M
thread-cache-size   = 16
thread-concurrency  = 24
query-cache-size    = 4096M
query-cache-limit   = 4M
tmp-table-size      = 128M
log_warnings
innodb_file_per_table=1
```

```

innodb_file_io_threads=4
innodb_open_files=2048
innodb_buffer_pool_size = 10G
innodb_additional_mem_pool=16M
innodb_thread_concurrency = 16
innodb_max_dirty_pages_pct = 90
log-bin=mysql-bin
binlog_format=mixed
server-id      = 1
[mysqldump]
quick
max_allowed_packet = 16M
[mysql]
no-auto-rehash
[myisamchk]
key_buffer_size = 20M
sort_buffer_size = 20M
read_buffer = 2M
write_buffer = 2M
[mysqlhotcopy]
interactive-timeout

```

12.7 其他方面

选用更好的硬件配置和快速的磁盘配置等工作应该是在规划之前进行的，如选用 raid10、SSD 固态硬盘、多核 CPU、大容量内存等硬件优化。

通常，在优化软件无效的情况下，换更好的硬件会有非常明显的效果，但大多数情况下，我们还是以优化软件为主。

对于特大型环境，可能由于 Zabbix 本身对某些方面的性能并未考虑周全，需要从代码级别进行二次开发优化。当然，这要根据自己的需求来改进。

第 13 章 Zabbix API 的使用

利用 Zabbix 的 API 功能可以很方便地通过其他程序调用 Zabbix, 从而实现灵活的扩展 Zabbix 方式。本章讲解了 Zabbix API 的使用, 对部分 Zabbix API 进行了分析。希望通过本章的学习, 能让读者对 Zabbix 的 API 有更深入的认识。

13.1 Zabbix API 简介

Zabbix API 具有重要的功能, 为第三方调用 Zabbix、批量操作提供可编程接口, 从而轻松地用于自己的业务系统, 将 Zabbix 监控系统与运维系统相集成。

Zabbix API 是基于前端 HTTP 协议实现的, 也就是可以通过 HTTP 请求实现的 API。API 数据传输采用 JSON RPC 协议。

由于 Zabbix 的 Web 前端是用 PHP 语言编写的, 而 PHP 的性能和相关配置参数有极大的关系。因此, 如果在大型的环境中使用, 可以对 PHP 进行负载均衡, 例如, 开启 PHP 多进程等方式来解决负载问题。除了对服务器本身进行优化外, 尽量减少对 API 的调用也是集成第三方系统应该遵循的一个原则。

在 Zabbix 2.2 中, 所有的 API 都有对应的官方文档和详细说明, 网址为 <https://www.zabbix.com/documentation/2.2/manual/api/reference>, 这里有所有的 API 用法。

13.2 JSON-RPC

既然 Zabbix API 采用 JSON-RPC 协议传输数据, 那么我们先来了解一下这个协议。

JSON-RPC 是基于 JSON 的跨语言远程调用协议, 比 XML-RPC、Webservice 等基于文本的协议传输数据量要小; 相比 Hessian、Java-RPC 等二进制协议更便于调试、实现、扩展, 是非常优秀的一种远程调用协议。目前主流语言都已有 JSON-RPC 的实现框架, Java 语言中较好的 JSON-RPC 实现框架有 Jsonrpc4j、Jpoxxy、JSON-RPC, 其中, Jsonrpc4j 既可独立使用, 又可与 Spring 无缝结合, 比较适用于基于 Spring 的项目开发。

1. JSON-RPC 协议描述

JSON-RPC 协议非常简单, 发起远程调用时向服务器端传输数据的格式如下。

```
{ "method": "getid", "params": ["arg"], "id": 1}
```

参数说明如下。

- method: 调用的方法名。
- params: 方法传入的参数，若无参数，则传入[]。
- id: 调用标识符，用于标识一次远程调用过程。

服务器收到 JSON-RPC 的调用请求，然后处理该请求的方法调用，并将方法调用处理后的结果响应给调用方。返回的数据格式如下。

```
{
  "result":      "id is 000",
  "error":      null,
  "id":         1
}
```

参数说明如下。

- result: 方法返回值，若无返回值或调用出错，则返回null。
- error: 调用时错误，无错误则返回null。
- id: 调用标识符，与调用方传入的标识符一致。

以上就是 JSON-RPC 协议规范的内容，它非常简单、小巧，且便于用各种语言实现。

2. Zabbix API 支持的数据类型

Zabbix API 支持的数据类型如表 13-1 所示。

表 13-1

类 型	说 明
bool	布尔值 true 或者 false
flag	当该值不等于空或者 false 时，被认为是 true
integer	整数
float	浮点数
string	文本字符串
timestamp	UNIX 时间戳
array	数组
object	关联数组
query	可以是一个数值，也可以是部分参数。 extend: 返回所有的对象值; count: 返回值的数量

在使用 get 方法的时候，支持的参数如表 13-2 所示。

表 13-2

参 数	类 型	描 述
countOutput	flag	返回结果的个数，而非实际的数据
editable	boolean	如果设置为 true，用户可对返回的对象进行写操作，默认为 false
excludeSearch	flag	返回不匹配给定参数的结果
filter	object	返回过滤后的结果，参数的值可以为一个数组或者是单个值，text 字段不能使用此参数
limit	integer	限制返回结果的数量
nodeids	string/array	返回给定节点的对象信息
output	query	返回对象的属性。默认为 extend
preservekeys	flag	返回以 ID 为键的数组
search	object	搜索匹配的字符串，仅用于字符和文本字段
searchByAny	boolean	如果设置为 true，则返回 filter 或 search 字段中所有的值。默认为 false
searchWildcardsEnabled	boolean	如果设置为 true，允许使用 “*” 作为搜索参数的通配符。默认为 false
sortfield	string/array	对给定的参数属性进行排序
sortorder	string/array	排序。ASC 为升序排列，DESC 为降序排列
startSearch	flag	搜索以某个参数开始的结果

3. Zabbix 2.0 和 Zabbix 2.2 的 API 改进

对于 Zabbix 2.0 和 Zabbix 2.2 的 API 改动，请参考文档：

https://www.zabbix.com/documentation/2.2/manual/api/changes_2.0_2.2

另外，API 的变化可参考如下地址中的文档：

<https://github.com/itnihao/zabbix-book/tree/master/17-chapter>

4. Zabbix API 代码的路径

Zabbix API 代码的路径如图 13-1 所示。

```
[root@www zabbix]# ls ap
api/
[root@www zabbix]# ls ap
```

图 13-1

前端 PHP 的代码中有一个 api_jsonrpc.php 文件，是 API 的入口文件。调用时的 URL 是 http://x.x.x.x/api_jsonrpc.php。

API 目录下面是 API 实现的代码。

13.3 Zabbix API 的使用流程

13.3.1 使用 API 的基本步骤

使用 API 的基本步骤如下。

① 连接 `http://x.x.x.x/api_jsonrpc.php`, 提供用户名和密码, 并标识 HTTP 头部 `Content-Type:"application/json"`, HTTP 方法为 `post`。

② 获取 `SESSIONID`。

③ 通过 `SESSIONID` 建立后续的连接。

④ 提交 `POST` 数据, 格式为 `JSON`, 其中放对应的方法, 获取需要的数据。

13.3.2 如何使用官方文档获取帮助

Zabbix API 的使用主要是参考官方文档, 这里我们要学习的是如何使用官方文档。例如, 想要获取 Host 的信息, 可查看官方文档, 地址如下。

<https://www.zabbix.com/documentation/2.2/manual/api/reference/host/get>

官方文档以参数列表的形式显示, 一些例子已经非常详细地进行了说明。

下面看一个官方的示例, 如图 13-2 所示。

Retrieving data by name

Retrieve all data about two hosts named "Zabbix server" and "Linux server".

Request:

```
{
  "jsonrpc": "2.0",
  "method": "host.get",
  "params": {
    "output": "extend",
    "filter": {
      "host": [
        "Zabbix server",
        "Linux server"
      ]
    }
  },
  "auth": "038e1d7b1735c6a5438ee9aae095879e",
  "id": 1
}
```

图 13-2

其中, `params` 可选的参数有很多, 如图 13-3 所示。

参数	类型	描述
groupids	string/array	Return only hosts that belong to
applicationids	string/array	Return only hosts that have the
dserviceids	string/array	Return only hosts that are relate
graphids	string/array	Return only hosts that have the
hostids	string/array	Return only hosts with the given
httpstestsids	string/array	Return only hosts that have the
interfaceids	string/array	Return only hosts that use the gi
itemids	string/array	Return only hosts that have the
maintenanceids	string/array	Return only hosts that are affect
monitored_hosts	flag	Return only monitored hosts.
proxy_hosts	flag	Return only proxies.
proxyids	string/array	Return only hosts that are monito

图 13-3

我们可以只对部分想要用的参数进行筛选，例如，上面的例子中，只用了 output 和 filter 这两个参数，如果还需要其他参数，则可以直接添加后使用。

13.3.3 用 CURL 模拟 API 的使用

下面用 CURL 浏览器来模拟 API 的使用过程。

1) 获取 Session 的语句如下。

```
shell#curl -i -X POST -H 'Content-Type:application/json' -d '{"
jsonrpc": "2.0","method":"user.authenticate","params":{"user":"Admin
","password":"zabbix"},"auth": null,"id":0}' http://192.168.0.200:8
090/zabbix/api_jsonrpc.php
```

得到的结果如下。

```
{
  "jsonrpc": "2.0",
  "result": "0d16a7b7cf3ed8394e020f54ffd48224",
  "id": 0
}
```

详细的 HTTP 头部如图 13-4 所示。

```
auth": null,"id":0}' http://192.168.0.200:8090/zabbix/api_jsonrpc.php
HTTP/1.1 200 OK
Date: Mon, 16 Dec 2013 04:42:50 GMT
Server: Apache
X-Powered-By: PHP/5.3.3
Content-Length: 68
Connection: close
Content-Type: application/json

{"jsonrpc": "2.0", "result": "0d16a7b7cf3ed8394e020f54ffd48224", "id": 0}
```

图 13-4

查询 MySQL 数据库可以看到，SESSIONID 已经存在于 zabbix.sessions 表中，如图 13-5 所示。

```
mysql> select * from zabbix.sessions;
```

sessionid	userid	lastaccess	status
043fe9e00e209f86a81c0c4ddaf9bd92	1	1387163963	0
05c5f95d1fe4739ad54ac73d5a65f995	1	1387164206	0
07a71c067223b52704454f48f6b53a58	1	1387169180	0
0d16a7b7cf3ed8394e020f54fffd48224	1	1387168970	0
1d55e205122ab60234d7ab12d3ae70e74	1	1387161490	0
243185f1a3103608329dbcca84145c4c	1	1387161335	0
2c0cc66f7127c7f89fd6ed483887a0bf	1	1387162713	0
3bb808fa03b567e55af2d5f946ce9f88	1	1387161853	0

图 13-5

2) 用 Session 请求去调用 API 的 host.get 方法的语句如下。

```
shell#curl -i -X POST -H 'Content-Type: application/json' -d '{
  "jsonrpc": "2.0", "method": "host.get", "params": {"output": "extend", "filter": {"host": ""}}, "auth": "0d16a7b7cf3ed8394e020f54fffd48224", "id": 1}'
  http://192.168.0.200:8090/zabbix/api_jsonrpc.php
```

得到的结果如图 13-6 所示。

```
[root@www zabbix]# curl -i -X POST -H 'Content-Type: application/json' -d '{"jsonrpc": "2.0", "method": "host.get", "params": {"output": "extend", "filter": {"host": ""}}, "auth": "0d16a7b7cf3ed8394e020f54fffd48224", "id": 1}' http://192.168.0.200:8090/zabbix/api_jsonrpc.php
HTTP/1.1 200 OK
Date: Mon, 16 Dec 2013 04:50:16 GMT
Server: Apache
X-Powered-By: PHP/5.3.3
Content-Length: 654
Connection: close
Content-Type: application/json

{"jsonrpc": "2.0", "result": [{"maintenances": [], "hostid": "10084", "proxy_hostid": "0", "errors_from": "0", "lastaccess": "0", "ipmi_authtype": "-1", "ipmi_privilege": "2", "ipmi_user": "root", "ipmi_disable_until": "0", "snmp_available": "0", "maintenance_status": "0", "ipmi_error": "0", "snmp_error": "0", "imx_disable_until": "0", "imx_available": "0"}], "id": 1}
```

图 13-6

注意，下面的格式是为了方便查看，对结果进行了排版。

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "maintenances": [],
      "hostid": "10084",
      "proxy_hostid": "0",
      "host": "Zabbix server",
      "status": "0",
      "disable_until": "0",
      "error": "",
      "available": "1",
      "errors_from": "0",
      "lastaccess": "0",
      "ipmi_authtype": "-1",
      "ipmi_privilege": "2",
      "ipmi_username": "",
      "ipmi_password": "",
      "ipmi_disable_until": "0",
      "ipmi_available": "0",
      "snmp_disable_until": "0",
      "snmp_available": "0",
      "maintenanceid": "0"
    }
  ],
  "id": 1
}
```

```

        "maintenance_status": "0",
        "maintenance_type": "0",
        "maintenance_from": "0",
        "ipmi_errors_from": "0",
        "snmp_errors_from": "0",
        "ipmi_error": "",
        "snmp_error": "",
        "jmx_disable_until": "0",
        "jmx_available": "0",
        "jmx_errors_from": "0",
        "jmx_error": "",
        "name": "Zabbix server"}
    ],
    "id": 1}

```

上面的结果中显示了如何使用 API，使我们对 API 有了一个宏观的认识。下面对 API 进行详细学习。

13.3.4 HTTP 头部 Content-Type 设置

HTTP 头部 Content-Type 必须设置为 application/json，否则，会提示状态码为 412 的错误（如果服务器没有满足请求者在请求中设置的其中一个前提条件，就会返回此错误代码），如图 13-7 所示。

```

[root@www zabbix]# curl -i -X POST -d '{"jsonrpc":"2.0","method":"zabbix.api.jsonrpc.ping","id":1}' http://192.168.0.200:8090/zabbix/api_jsonrpc.php
HTTP/1.0 412 Precondition Failed
Date: Mon, 16 Dec 2013 05:05:22 GMT
Server: Apache
X-Powered-By: PHP/5.3.3
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8

```

图 13-7

13.3.5 关于用户认证

认证成功的提示内容如下。

```

{
    "jsonrpc": "2.0",
    "result": "0d16a7b7cf3ed8394e020f54ffd48224",
    "id": 0
}

```

认证失败的提示内容如下。

```

{
    "jsonrpc": "2.0",
    "error": {
        "code": -32602,

```



```

        "message": "Invalid params.",
        "data": "Login name or password is incorrect."
    },
    "id": 0
}

```

所以，在调用 API 的时候，对是否认证成功需要进行判断，以给出友好的信息提示，便于排除错误。

13.3.6 获取主机信息（用 Python 写的示例）

关于获取主机信息的 API 方法，请参看以下地址。

<https://www.zabbix.com/documentation/2.2/manual/api/reference/host/get>

继续前面的例子，下面用 Python 语言来实现，具体代码如下。

```

#!/usr/bin/env python
#coding=utf-8

#导入模块，urllib2是一个模拟浏览器HTTP方法的模块
import json
import urllib2
import sys
from urllib2 import Request, urlopen, URLError, HTTPError

#url and url header
#zabbix的API地址、用户名、密码，这里修改为实际的参数
zabbix_url="http://192.168.0.200:8090/zabbix/api_jsonrpc.php"
zabbix_header = {"Content-Type":"application/json"}
zabbix_user   = "admin"
zabbix_pass   = "zabbix"
auth_code     = ""

#auth user and password
#用户认证信息的部分，最终的目的是得到一个SESSIONID
#下面是生成一个JSON格式的数据：用户名和密码
auth_data = json.dumps(
    {
        "jsonrpc": "2.0",
        "method": "user.login",
        "params":
            {
                "user": zabbix_user,
                "password": zabbix_pass
            },
        "id": 0
    })

# create request object
request = urllib2.Request(zabbix_url, auth_data)
for key in zabbix_header:
    request.add_header(key, zabbix_header[key])

```

```

#认证和获取SESSION ID
try:
    result = urllib2.urlopen(request)
#对于认证出错的处理
except HTTPError, e:
    print 'The server couldn\'t fulfill the request, Error code: ',
e.code
except URLError, e:
    print 'We failed to reach a server.Reason: ', e.reason
else:
    response=json.loads(result.read())
    result.close()
...
#如果访问成功或者失败，这里的数据会显示如下
    sucess result:
        {"jsonrpc":"2.0",
         "result":"0d225d8d2a058625f814f3a0749cd218",
         #result后面的值是SESSIONID，每次访问都会发生变化
         "id":0}
    error result:
        {'code': -32602,
         'data': 'Login name or password is incorrect.',
         'message': 'Invalid params.'}
...
#判断SESSIONID是否在返回的数据中
    if 'result' in response:
        auth_code=response['result']
    else:
        print response['error']['data']

# request json
json_data={
    "method":"host.get",
    "params":{
        "output": "extend",
    }
}
json_base={
    "jsonrpc":"2.0",
    "auth":auth_code,
    "id":1
}
json_data.update(json_base)
#用得到的SESSIONID去验证，获取主机的信息（用http.get方法）
if len(auth_code) == 0:
    sys.exit(1)
if len(auth_code) != 0:
    get_host_data = json.dumps( json_data)
    # create request object
    request = urllib2.Request(zabbix_url,get_host_data)
    for key in zabbix_header:

```



```

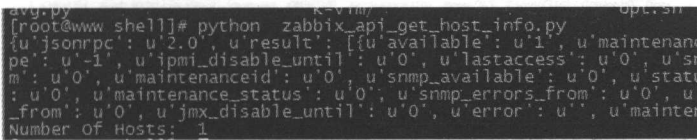
request.add_header(key, zabbix_header[key])

# get host list
try:
    result = urllib2.urlopen(request)
except URLError as e:
    if hasattr(e, 'reason'):
        print 'We failed to reach a server.'
        print 'Reason: ', e.reason
    elif hasattr(e, 'code'):
        print 'The server could not fulfill the request.'
        print 'Error code: ', e.code
else:
    response = json.loads(result.read())
    result.close()

#将所有的主机信息显示出来
print response
#显示主机的个数
print "Number Of Hosts: ", len(response['result'])

```

将以上代码保存为 `zabbix_api_get_host_info.py`，注意修改用户名、密码和 url 为自己实际的环境，运行结果如图 13-8 所示。



```

[root@www shell]# python zabbix_api_get_host_info.py
{'jsonrpc': '2.0', 'uresult': [{'available': '1', 'maintenance': '0', 'ipmi_disable_until': '0', 'lastaccess': '0', 'snmp_available': '0', 'maintenance_status': '0', 'snmp_errors_from': '0', 'jmx_disable_until': '0', 'error': '0', 'maintenance_status': '0'}], 'Number Of Hosts: 1'}

```

图 13-8

13.3.7 添加 Host

添加 Host 的方法请参考官方文档，网址如下：

<https://www.zabbix.com/documentation/2.2/manual/api/reference/host/create>

看到 `host.create` 的用法如下：

```

{
  'auth': '038e1d7b1735c6a5436ee9eae095879e',
  'id': 1,
  'jsonrpc': '2.0',
  'method': 'host.create',
  'params': {'groups': [{'groupid': '50'}],
    'host': 'Linux server',
    'interfaces': [{'dns': '',
      'ip': '192.168.3.1',
      'main': 1,
      'port': '10050',
      'type': 1,
      'useip': 1}],

```

```
'inventory': {'macaddress_a': '01234', 'macaddress_b':  
'56768'},  
  'templates': [{'templateid': '20045'}]}
```

上述语句表示添加 IP 为 192.168.3.1、hostname 为 Linux Server 的这台主机，将分组放在 id 为 50 的主机组中，引用 id 为 20045 的模板。

由于官方示例中的部分参数并不是实际环境的参数，所以在 API 中使用时需要用实际存在的参数。

在 MySQL 数据库中查看模板 id (严格地说，获取模板的 ID 也应该通过 API，但此处更多的是展示 API 的用法，以及对应数据库中的关系)，确保存在此 ID，这里选用 Template OS Linux 模板，看到其 id 为 10001，故 host.create 中的 templateid 参数为 10001，如图 13-9 所示。

```
mysql> select * from hosts where host="Template OS Linux"\G;  
1. row  
  hostid: 10001  
 proxy_hostid: NULL  
   host: Template OS Linux  
  status: 3  
disable_until: 0
```

图 13-9

Groupid 在这里归类到 id 为 12 的组中，如图 13-10 所示。

因此，API 示例中的 json_data 将由图 13-11 中的语句变为下面的语句。

```
mysql> select * from groups;  
+-----+-----+-----+-----+  
|groupid|name      |internal|flags|  
+-----+-----+-----+-----+  
|1|templates|0|0|  
|2|linux servers|0|0|  
|4|zabbix servers|0|0|  
|5|discovered hosts|1|0|  
|6|virtual machines|0|0|  
|7|hypervisors|0|0|  
|8|business manager group1|0|0|  
|9|business manager group2|0|0|  
|10|business manager group3|0|0|  
|11|devops|0|0|  
|12|dev web app group1|0|0|  
|13|dev web app group2|0|0|  
|14|dev web app group3|0|0|  
+-----+-----+-----+-----+
```

图 13-10

```
# request json  
json_data={  
  "method":"host.get",  
  "params":{  
    "output":"extend",  
  }  
}
```

图 13-11

```
{  
  'method': 'host.create',  
  'params': {'groups': [{'groupid': '12'}],  
    'host': 'Web Linux server',
```

```
'interfaces': [{'dns': '',
                'ip': '192.168.8.1',
                'main': 1,
                'port': '10050',
                'type': 1,
                'useip': 1}],
'inventory': {'macaddress_a': '01234', 'macaddress_b':
'56768'},
'templates': [{'templateid': '10001'}]}
```

运行代码，将会看到如图 13-12 所示的提示。

```
[root@mysql-master zabbix-api]# python zabbixhostcreate.py
{'jsonrpc': '2.0', 'params': {'templates': [{'templateid': '10001'}], 'host': 'web linux server', 'interfaces': [{'ip': '192.168.8.1', 'port': '10050'}], 'groups': [{'groupid': '12'}], 'inventory': {'macaddress_b': '56768', 'macaddress_a': '01234', 'id': 1}}, 'id': 1}
{'jsonrpc': '2.0', 'result': {'u'hostids': [u'10113'], 'u'id': 1}}
Number of Hosts: 1
```

图 13-12

在 Web 界面中可以看到刚添加的主机，如图 13-13 所示。

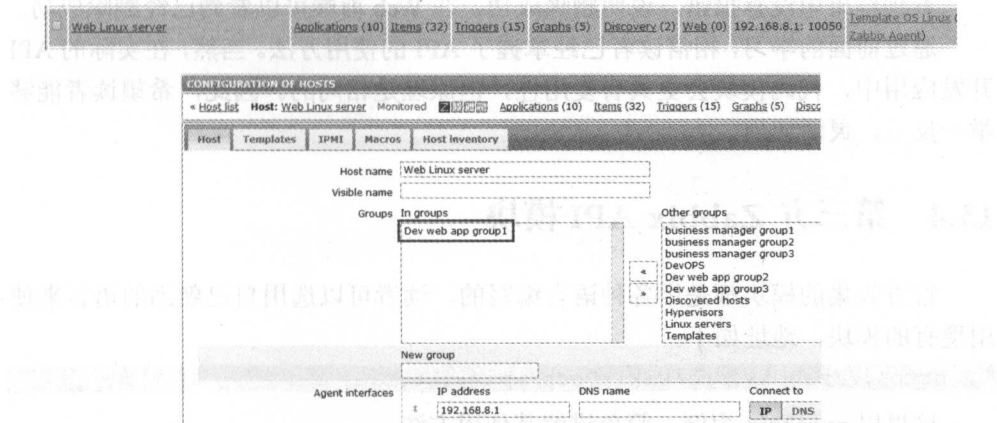


图 13-13

13.3.8 删除 Host

关于删除 Host 的方法，先参考官方文档，地址如下。

<https://www.zabbix.com/documentation/2.2/manual/api/reference/host/delete>

官方给出的示例如下。

```
{'auth': '038e1d7b1735c6a5436ee9eae095879e',
'id': 1,
'jsonrpc': '2.0',
'method': 'host.delete',
'params': ['13', '32']}
```

上述语句表示删除 hostid 为 13 和 32 的两台主机。

通过运行 `host.get` (13.3.6 节中的脚本), 知道了 `hostid` 为 10113。

```
hostid': u'10113', u'name': u'Web Linux server', u'jmx_errors_from': u'0'
```

`json_data` 将改变为如图 13-14 所示的语句。

```
# request json
json_data={
    "method": "host.get",
    "params": ['10113']
}
```

图 13-14

运行脚本, 如图 13-15 所示。

```
[root@mysql-master zabbix-api]# ./zabbixhostdelete.py
{u'jsonrpc': u'2.0', u'result': {u'hostids': [u'10113'], u'id': 1}
Number of Hosts: 1
[root@mysql-master zabbix-api]#
```

图 13-15

看到结果中没有报错, 说明删除成功。在 Web 前端可以看到已经删除成功。

通过前面的学习, 相信读者已经掌握了 API 的使用方法。当然, 在实际的 API 开发应用中, 代码模块会更具有复用性, 但原理是相同的。因此, 希望读者能够举一反三, 灵活应用。

13.4 第三方 Zabbix API 模块

官方收集的模块都是用各种语言编写的, 读者可以选用自己熟悉的语言来使用现有的模块, 地址如下。

<https://zabbix.org/wiki/Docs/api/libraries>

这里以 `pyzabbix` 为例, 简单讲解其使用方法。

`shell#pip install pyzabbix` #如图13-16所示

```
in [1]: from pyzabbix import ZabbixAPI
in [2]: zapi = ZabbixAPI("http://zabbix-gui.itnihao.com")
in [3]: zapi.login("admin", "zabbix")
in [4]: print "Connected to Zabbix API Version %s" % zapi.api_version()
         print("Connected to zabbix API Version %s" % zapi.api_version())
Connected to Zabbix API Version 2.2.2
in [5]: for h in zapi.host.get(extendoutput=True):
         print h['hostid']

10084
10106
10107
10108
10109
10110
10111
10112
```

图 13-16

更多的例子见 <https://github.com/lukecyca/pyzabbix/tree/master/example>。

本章讲解了 Zabbix API 的使用方法，并对照官方文档和实际应用做了说明，让读者理解 API 的使用方法，但未做更多实际案例的深入分析，这需要读者根据实际应用，选择合适的 API 模块，或者是自己写 API 模块，基于此，读者可以二次定制 Zabbix 所需的应用界面和编程接口，更好地与第三方应用系统进行结合，从而极大地扩展 Zabbix，提高应用的灵活性。

名称	描述	数据类型	是否必填	备注
主机名称	主机的名称	字符串	是	
主机 IP	主机的 IP 地址	字符串	是	
主机组	主机所属的组	字符串	否	
主机模板	主机所属的模板	字符串	否	
主机代理	主机使用的代理	字符串	否	
主机版本	主机的版本	字符串	否	
主机状态	主机的状态	字符串	否	
主机备注	主机的备注	字符串	否	

第 14 章 使用 Zabbix 协议

前面我们学习了如何使用 Zabbix API，在某些场合中，我们更希望使用自己开发的客户端，即在自己的业务程序中内嵌 Zabbix 客户端，用自己写的客户端发送数据给 Zabbix-Server。那么，这个需求是否能够满足呢？答案是肯定的。Zabbix-Agent 和 Zabbix-Server 之间的通信是采用 Zabbix 协议来实现的，在 7.4 节中，我们学习了 Zabbix-Sender 这种主动的方式发送数据给 Zabbix Server，因此，我们完全可以自己写一个 Zabbix-Sender 内置到业务程序中，从而避免在系统中安装 Zabbix 官方提供的客户端程序。

当我们掌握了 Zabbix 协议后，无论是进行 Zabbix 的二次开发、改变原有的代码，还是写一个新的客户端程序，都具有极大的意义，将会极大地扩展 Zabbix 应用的范围和可编程性。例如，可以将 Zabbix 集成到嵌入式系统，甚至集成到其他硬件系统。

在本章中，将介绍 Zabbix 的基本协议，如 Zabbix-Agent、Zabbix-Sender 和 Zabbix-Get 协议，其他协议请读者参考官方文档。

本章所有的代码链接地址为：

<https://github.com/itnihao/zabbix-book/blob/master/14-chapter>

14.1 Zabbix 协议概述

Zabbix 协议是 Zabbix 各程序间通信的准则，表 14-1 展示了各模块的协议和版本支持概况。

表 14-1

协 议	版 本 号					
	1.1	1.4	1.6	1.8	2.0	2.2
zabbix agent	支持	支持	支持	支持	支持	支持
zabbix sender	\	\	\	支持	支持	支持
zabbix proxy	\	\	支持	支持	支持	支持
zabbix Java gateway	\	\	\	\	支持	支持
zabbix global scripts	\	\	>1.69	\	支持	支持
zabbix queue	\	\	\	\	\	支持

注意：读者可以在官方地址 <https://www.zabbix.org/wiki/Docs/protocols> 中找到此表。

在 Zabbix 各程序间的通信中，其协议传输数据的格式为 JSON（JSON 的相关资料，请读者参考网络资料），在 Zabbix 中的 JSON 数据输出时，某些 JSON 数据不便于阅读，读者可以用 JSON 格式转换的在线网站（如 <http://jsonlint.com>）进行转换，或采用 Python 库进行格式化输出，具体如下。

```
shell# zabbix_get -s 127.0.0.1 -k net.if.discovery | python -mjson.tool
{
  "data": [
    {
      "#{IFNAME}": "lo"
    },
    {
      "#{IFNAME}": "eth0"
    }
  ]
}
```

注：关于 python -mjson.tool 的用法，参考 <https://docs.python.org/2/library/json.html>。

14.2 Zabbix-Sender 协议

在学习 Zabbix-Sender 协议之前，先学习 Zabbix 协议中的一些规则，下面展示的是如何获取 Zabbix 的信息，即 Zabbix-Get 的功能。

Zabbix-Server 建立 TCP 连接到 10050 端口，请求具体的某个 key 值，Zabbix-Agent 对请求进行数据的响应，下面用 Telnet 来进行模拟测试。

```
shell# telnet 127.0.0.1 10050
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
agent.version
ZBXD0r2.2.2Connection closed by foreign host.
```

上述语句中，请求的 key 是 agent.version，响应的数据是 ZBXD 0 2.2.0rc2，用 Zabbix-Get 程序获取到的数据如下。

```
shell# zabbix_get -s 127.0.0.1 -k agent.version
2.2.2
```

Agent 响应的数据中有一个头部信息 ZBXD，如图 14-1 所示。

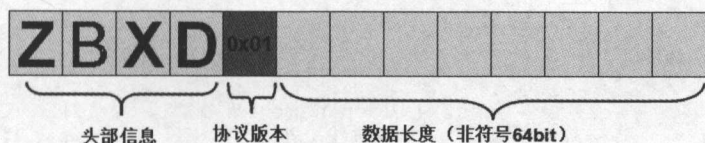


图 14-1

Zabbix Agent 的数据发送格式如下。

<HEADER><DATALEN><DATA>

解释如下。

- <HEADER>: "ZBXD\x01" (5字节)。
- <DATALEN>: 数据长度 (8字节), 1将被格式化为01/00/00/00/00/00/00/00 (十六进制数8字节, 64位数字)。
- <DATA>: 大小有限制。为了不耗尽Zabbix-Server的内存, 在Zabbix 2.2.0~Zabbix 2.2.2中, Zabbix-Server将单个数据的接收大小限制为64MB, Zabbix 2.2.0以前的版本为128MB, 在Zabbix 2.0.3版本之前对大小无任何限制。在Zabbix 2.2.3中, 改为128MB的大小限制, 在64MB限制的版本中将会丢弃超过64MB大小限制的数据。

图 14-2 更加详尽地解释了上面的数据格式。

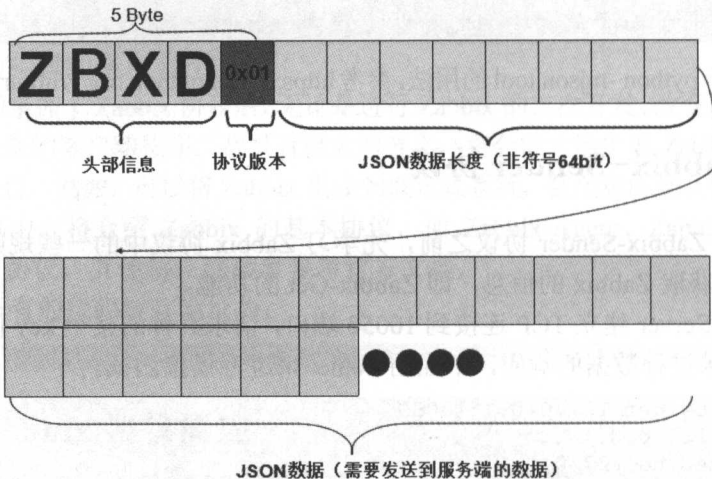


图 14-2

下面分别介绍 Sender、Get 和 Agent 协议。

14.2.1 Sender 数据发送

Zabbix-Sender 发送数据时, 其数据格式为 JSON, 数据的三要素是: host、key、value, 具体内容如下。

```
{
  "request": "sender data",
  "data": [
    {
      "host": "Host name 1",
      "key": "item_key",
      "value": "33"
    },
    {
```

```

        "host": "Host name 2",
        "key": "item_key",
        "value": "55"
    }
]
}

```

14.2.2 Server 对数据响应的处理

Zabbix-Server 对请求的数据进行响应，其响应的数据格式如下。

```

{
    "response": "success",
    "info": "Processed 1 Failed 1 Total 2 Seconds spent 0.000253"
}

```

如果发送的数据中还包含时间戳，格式如下。

```

{
    "request": "sender data",
    "data": [
        {
            "host": "Host name 1",
            "key": "item_key",
            "value": "33",
            "clock": 1381482894
        },
        {
            "host": "Host name 2",
            "key": "item_key",
            "value": "55",
            "clock": 1381482894
        }
    ],
    "clock": 1381482905
}

```

14.2.3 Zabbix-Sender 的实例

下面的示例展示了如何用 Python 来编写一个 Zabbix-Sender 的客户端程序。

```

#!/usr/bin/env python
# -*- coding: utf-8 -*-
import socket
import struct
import simplejson

class ZabbixSender:

    zbx_header = 'ZBXD'
    zbx_version = 1
    zbx_sender_data = {'request': 'sender data', 'data': []}
    send_data = ''

```

```

def __init__(self, server_host, server_port = 10051):
    self.server_ip = socket.gethostbyname(server_host)
    self.server_port = server_port

def AddData(self, host, key, value, clock = None):
    add_data = {u'host': host, u'key': key, u'value': value}

    if clock != None:
        add_data[u'clock'] = clock
    self.zbx_sender_data['data'].append(add_data)
    return self.zbx_sender_data

def ClearData(self):
    self.zbx_sender_data['data'] = []
    return self.zbx_sender_data

def __MakeSendData(self):
    zbx_sender_json = simplejson.dumps(self.zbx_sender_data, separators=(',', ':'), ensure_ascii=False).encode('utf-8')
    json_byte = len(zbx_sender_json)
    self.send_data = struct.pack("<4sBq" + str(json_byte)
+ "s", self.zbx_header, self.zbx_version, json_byte, zbx_sender_json)

def Send(self):
    self.__MakeSendData()
    so = socket.socket()
    so.connect((self.server_ip, self.server_port))
    wobj = so.makefile(u'wb')
    wobj.write(self.send_data)
    wobj.close()
    robj = so.makefile(u'rb')
    recv_data = robj.read()
    robj.close()
    so.close()
    tmp_data = struct.unpack("<4sBq" + str(len(recv_data)
- struct.calcsize("<4sBq"))) + "s", recv_data)
    recv_json = simplejson.loads(tmp_data[3])
    return recv_data

if __name__ == '__main__':
    sender = ZabbixSender(u'127.0.0.1')
    for num in range(0,2):
        sender.AddData(u'Zabbix server', u'test', u'sent data'
a' + str(num))
        res = sender.Send()
        print sender.send_data
        print res

```

在上面的代码中，有如下字段：

```
sender.AddData(u'Zabbix server', u'test', u'sent data' + str(num))
```

解释如下：

- Zabbix Server 为 host。
- test 为 key。
- sent data 为 value。

为了使上例中的客户端能够正常运行，需要在 Zabbix GUI 的 Zabbix-Server 主机上建立一个 Trapper 类型的 Items，其 key 为 test。

上例代码的主要步骤是 Sender 和 Server 建立 Socket 连接、发送 key 的数据（注意头部信息的正确性）、Server 对数据进行处理响应的一个过程，可参考官方文档，地址如下。

https://www.zabbix.org/wiki/Docs/protocols/zabbix_sender/2.0

下面是官方文档中 Java 的例子。

```
String report = buildJsonString(...);
writeMessage(stream, report.getBytes());

...
private String buildJsonString(String host, String item, String value)
{
    return
        "{"
        + "\"request\":\"sender data\", \"\n"
        + "\"data\":{\"\n"
        + "    {\n"
        + "        \"host\":\"" + host + "\",\n"
        + "        \"key\":\"" + item + "\",\n"
        + "        \"value\":\"" + value.replace("\\", "\\")
        + "\"}}}\n" ;
}

protected void writeMessage(OutputStream out, byte[] data) throws IOException {
    int length = data.length;

    out.write(new byte[] {
        'Z', 'B', 'X', 'D',
        '\1',
        (byte)(length & 0xFF),
        (byte)((length >> 8) & 0x00FF),
        (byte)((length >> 16) & 0x0000FF),
        (byte)((length >> 24) & 0x000000FF),
        '\0', '\0', '\0', '\0'});

    out.write(data);
}
```

Perl 的例子参考如下地址。

https://www.zabbix.org/wiki/Docs/protocols/zabbix_sender/1.1/perl_example

14.3 Zabbix-Get 协议

Zabbix-Get 的例子在本书相关的代码中，读者可从以下地址下载参考。

<https://github.com/itnihao/zabbix-book/blob/master/14-chapter/zabbix-get.py>

核心代码如下：

```
client_socket=None
try:
    client_socket = socket.create_connection((options.host, options.port), options.timeout)

    protocol = ZBXDPProtocol()
    protocol.send_value(client_socket, key)

    print(protocol.receive_value(client_socket))
except Exception as e:
    print("Unable to receive data from agent: {}".format(e))
```

用法如下：

```
shell# python zabbix-get.py -s 127.0.0.1 -p 10050 system.uname
Linux zabbix.itnihao.com 2.6.32-279.el6.x86_64 #1 SMP Wed Jun 13 18:24:36 EDT 2012 x86_64
```

14.4 Zabbix-Agent 协议

下面来看看 Zabbix-Agent 协议的实现过程。

对于被动模式的实现过程，请读者参考本书 8.3 节的内容，在这里主要讲解主动模式的实现过程。

1. Agent 启动的时候请求 Server

Zabbix-Agent 端请求 Zabbix-Server 端，其发送的数据格式如下。

```
{
    "request": "active checks",
    "host": "Host name"
}
```

2. Server 响应 Agent 请求

Zabbix-Server 端对请求进行响应，其数据格式如下。

```
{
    "response": "success",
    "data": [
        {
            "key": "log[/var/log/localmessages,@errors]",

```



```

        "delay":1,
        "lastlogsize":12169,
        "mtime":0}},
    "regexp":[
        {
            "name":"errors",
            "expression":"^error",
            "expression_type":3,
            "exp_delimiter":",",
            "case_sensitive":1}}]

```

3. Agent 发送数据给 Server

Zabbix-Agent 将数据发送给 Zabbix-Server 端，其发送的数据格式如下。

```

{
    "request":"agent data",
    "data":[
        {
            "host":"Server 115",
            "key":"log[\\var\\log\\localmessages]",
            "value":"Apr 29 07:26:13 zabbix-master Keepal
ived_vrrp[1854]: Netlink: filter function error",
            "lastlogsize":4315,
            "clock":1360314499,
            "ns":699351525}
        ],
        "clock":1360150949,
        "ns":412904960
    ]
}

```

4. Server 响应 Agent，返回状态结果

Zabbix-Server 对 Zabbix-Agent 发送过来的数据进行处理，并返回处理结果。

```

{
    "response":"success",
    "info":"Processed 3 Failed 0 Total 3 Seconds spent 0.000110"
}

```

响应异常的处理如下。

① 部分请求成功，部分失败的响应如下。

```

{
    "response":"success",
    "info":"Processed 2 Failed 1 Total 3 Seconds spent 0.000113"
}

```

② 无主动监控项的处理如下。

```

{
    "response":"success",
    "data":[]
}

```

③ 请求的主机不存在的语句如下。

```
{
    "response": "failed",
    "info": "host [Host name] not found"
}
```

④ 主机未监控的语句如下。

```
{
    "response": "failed",
    "info": "host [Host name] not monitored"
}
```

5. Agent 协议的实例

用 Zabbix-Agent 协议编写的客户端程序在网络上已经有相关的开源项目，读者可参考如下链接。

```
https://github.com/ehl16/pyzbxagent
https://github.com/blin/zabby
https://github.com/nikicat/zabbix-agent-ng
```

第 15 章 定制 Zabbix 安装包

本章继续对 Zabbix 的自动化运维进行了扩展，通过对 Zabbix 安装包的定制，可以实现批量的安装和配置，这是部署大规模环境时必须掌握的技巧。本章通过一个定制 Zabbix 安装包的项目，让读者理解自动化运维中的安装配置。

15.1 为什么要定制安装包

Zabbix 的安装部署可以采用源码包完成，也可以采用官方的 RPM 包完成，但在更多的情况下，我们有必要对 Zabbix 的安装包进行定制。我们需要修改自定义 key 的配置文件、脚本文件，以及 conf 文件的默认参数，然后进行打包。对于这样的需求，我们可以通过配置管理工具进行统一的推送维护，但在某些情况下，我们更希望这些脚本在安装的时候就自动完成配置的工作。另外，官方提供的 RPM 依赖包固定，不太适合自己的安装需求，如官方的 RPM 包依赖于 APACHE，而不支持 Nginx，在这种需求下，对安装包进行二次定制很有必要。定制后的安装包有利于统一维护管理和自动化配置。

对于安装包的定制，在不同的平台，其软件包的封装形式会不同，本章将以 RPM 包的定制为例进行介绍（DEB 软件包管理不涉及）。

关于 Rpmbuild 的更多资料，请读者参考相关资料和书籍，这里不再详述。

15.2 如何定制安装包

有关定制 RPM 包（也就是 Rpmbuild），对运维人员来说，掌握软件包定制的技术对以后的工作将会产生很大影响，尤其是在自动化运维的配置管理方面。

在这里，笔者做了一个关于 Zabbix 的 RPM 包定制的项目，项目名字为 zabbix-rpm，代码托管在 github 中，网址如下：

```
https://github.com/itnihao/zabbix-rpm
```

这个项目的意义如下。

- 快速安装，标准配置，几分钟之内即可构建 Zabbix 系统，解决新手安装问题，获得很好的体验，能快速学习并上手。

- 大规模批量安装，提供 RPM 包定制的规范，可以作为参考案例。
- 提供了 SRPM 包，可以基于二次定制。
- 集成更多的脚本模板，方便大家使用。

如何重新打包 RPM 呢？

登录到 Linux 系统（这里以 RHEL 6.X、CentOS 6.X 系统为例）。

```
shell# yum install rpm-build git
shell# useradd admin
shell# su - admin
shell# mkdir -pv rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS}
shell# echo "%_topdir /home/admin/rpmbuild" > ~/.rpmmacros
```

在该项目 <https://github.com/itnihao/zabbix-rpm> 中找到对应的版本，如 Zabbix 2.2.2，下载 zabbix-2.2.2-0.el6.zbx.src.rpm 源码 RPM 软件包。

```
shell# rpm2cpio zabbix-2.2.2-0.el6.zbx.src.rpm | cpio -div
#解压源码rpm包
```

解压出来的文件如下：

- cmdline-jmxclient-0.10.3.jar
- zabbix-2.2.2.tar.gz
- zabbix-apache-web.conf
- zabbix-java-gateway.init
- zabbix-logrotate.in
- zabbix-nginx-web.conf
- zabbix2.2.2.spec
- zabbix_custom.tar.gz
- zabbix_java_gateway_cmd

准备构建 RPM 包，语句如下。

```
shell# mv zabbix2.2.2.spec /home/admin/rpmbuild/SPECS
shell# mv * /home/admin/rpmbuild/SOURCES
```

重新打包 RPM，语句如下。

```
shell# cd /home/admin/rpmbuild/SPECS
shell# rpmbuild -ba zabbix2.2.2.spec #此处会提示你需要依赖包，依次安装
```

在执行完上述命令后，会自动生成 RPM 包，读者可以参考该项目的 Readme 文档进行打包安装。

在 SPEC 文件中可以编写很多 Shell 命令，如下面的参数中，对 ServerActive、Server 等默认的参数进行了修改，这样在安装好之后，就无须进行二次配置，安装后即配置完毕。

```

sed -i \
-e 's|# PidFile=.*|PidFile=%{_localstatedir}/run/%{name}/zabbix_agentd.pid|g' \
-e 's|^LogFile=.*|LogFile=%{_localstatedir}/log/%{name}/zabbix_agentd.log|g' \
-e '/# UnsafeUserParameters=0/aUnsafeUserParameters=1\n' \
-e '/# Include.*zabbix_agentd.conf.d//aInclude=/etc/zabbix/zabbix_agentd.conf.d/\n'\
-e '/StartAgents=3/aStartAgents=5\n' \
-e 's|# LogFileSize=.*|LogFileSize=0|g' \
-e 's|Server=127.0.0.1|Server=127.0.0.1,10.10.10.1|g' \
-e 's|ServerActive=127.0.0.1|ServerActive=127.0.0.1:10051,10.10.10.1:10051|g' \
-e 's|# EnableRemoteCommands=0|EnableRemoteCommands=1|g' \
-e 's|# LogRemoteCommands=0|LogRemoteCommands=1|g' \
-e 's|LogFileSize=0|LogFileSize=10|g' \
-e 's|/usr/local|/usr|g' \
$RPM_BUILD_ROOT%{_sysconfdir}/%{name}/zabbix_agentd.conf

sed -i \
-e 's|/usr/local|/usr|g' \
-e '/# UnsafeUserParameters=0/aUnsafeUserParameters=1\n' \
-e 's|# Include=/usr/etc/zabbix_agentd.conf.d|Include=/etc/zabbix/zabbix_agentd.conf.d|g' \
$RPM_BUILD_ROOT%{_sysconfdir}/%{name}/zabbix_agent.conf

```

包安装完成后自动添加服务，代码如下。

```

%post agent
if [ $1 -eq 1 ]; then
sed -i "s@Hostname=Zabbix server@Hostname=$HOSTNAME@g" /etc/zabbix/zabbix_agentd.conf
getent group zabbix >/dev/null || groupadd -r zabbix
getent passwd zabbix >/dev/null || useradd -r -g zabbix -d %{_sharedstatedir}/zabbix -s /sbin/nologin -c "zabbix user" zabbix
/sbin/chkconfig zabbix-agent on
/sbin/service zabbix-agent start
chown root:zabbix /bin/netstat
chmod 4755 /bin/netstat
fi

```

更多的内容请读者参考对应版本的 SPEC 文件，如 Zabbix 2.2.2.spec，在这个文件中，读者可以看到更多关于定制 Zabbix 的 RPM 内容。

有关 DEB 打包的内容，网址如下。

https://www.zabbix.org/wiki/Docs/howto/rebuild_debs

第 16 章 大型分布式监控案例

本章通过一个大型的分布式监控环境,对 Zabbix 的大型环境进行了深入探讨,从自定义 RPM 包到架构设计,再到各个环节的优化配置,让读者进行深入的应用管理。

16.1 监控系统构建概述

前面我们学习了如何使用 Zabbix,包括很多使用技巧,部分章节中提到了大规模分布式监控应该注意的事项,本章通过一个实际的案例介绍在大规模环境中如何部署和运维一个监控系统。

在大型的 IT 架构环境中,IT 系统的组成通常是跨区域分布在多个省市的;跨节点,多个 IDC,而在中国,网络线路分为电信、联通,移动等多种复杂的网络;业务类型众多、系统构成复杂、业务需求多样,是这种大型环境的特点。在互联网业务中,新旧替代速度快是众所周知的,因此,监控系统要能满足业务不断变化的需求。

在这种环境中去构建监控系统,首先要做的事情是掌握全局信息,以及需要考虑未来的发展趋势。接下来是选择合适的技术方案,通常,这个技术方案既要能满足于当前,又要能满足不断增长的需求,所以合理的规划和预测,加上一定的理论和实践经验,才能设计一套比较完美的解决方案。

在大规模的监控系统中,需要考虑以下因素。

第一,分布式架构为首要考虑因素。要求系统架构具备分布式的设计,原则是将中心节点压力分散在各边缘节点,使其尽可能监控更多的设备。

第二,对数据存储的可扩展成了一个必须解决的问题。节点数量的众多给监控数据的存储带来了十分严峻的挑战,能否解决这个问题成了整个监控能否正常工作的前提条件。

第三,系统的高可用和健壮性、稳定可靠的系统架构、冗余的灾备,成为大型监控系统必须具备的条件。

第四,具备为其他业务系统提供 API 的能力,让它和其他业务系统的联系更为紧密。一个孤立的监控系统在大型环境中会对其他业务系统造成很大的麻烦,通常需要花费更多的精力去改造,使其为其他系统提供所需的数据。

第五,具备自动化功能。自动化是解决繁重体力劳动最有效的方式,未来

的运维一定更智能、更偏向于业务。以业务为核心，而不是仅仅解决系统的底层问题。

以上观点可以适用于任何监控系统的架构中，而不仅仅是本书中探讨的 Zabbix。

16.2 监控环境架构图

在图 16-1 中，我们可以看到监控系统和各业务系统紧密结合，通过分布式架构，使其能工作于复杂的环境中。

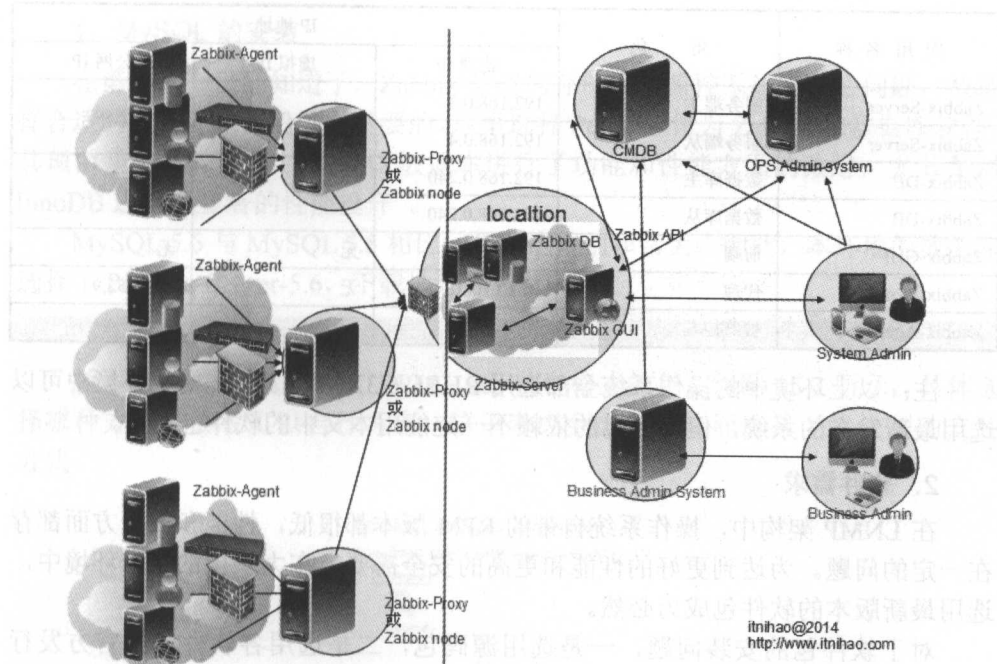


图 16-1

下面将详细介绍这个架构的实现过程。

16.3 架构实现的过程

16.3.1 硬件和软件需求

1. 硬件需求

硬件需求如表 16-1 所示。

表 16-1

应用名称	操作系统	CPU	内存	硬盘
Zabbix-Server	RHEL6.5_64	4 核心	DDR3 16GB	SATA 500GB×2 RAID1
Zabbix-DB	RHEL6.5_64	16 核心	DDR3 32GB	SAS 300GB×4 RAD1+0
Zabbix-GUI	RHEL6.5_64	4 核心	DDR3 4GB	SATA 500GB
Zabbix-Proxy	RHEL6.5_64	8 核心	DDR3 16GB	SAS 300GB×2 RAID1

IP 地址规划如表 16-2 所示。

表 16-2

应用名称	角 色	IP 地址		
		物理 IP	虚拟 IP	公网 IP
Zabbix-Server	服务端主	192.168.0.3	192.168.0.5	58.75.2.89
Zabbix-Server	服务端从	192.168.0.4		
Zabbix-DB	数据库主	192.168.0.240		无
Zabbix-DB	数据库从	192.168.0.240		
Zabbix-GUI	前端	192.168.0.2	无	无
Zabbix-Proxy	代理	10.10.10.2	无	61.61.52.9
Zabbix-Agent	被监控端	10.10.10.10	无	无

注：以上环境中的操作系统全部选用 RHEL 6.3X64，读者在构建环境中可以选用最新发布系统，但软件包的依赖不一定能用本文中的软件包版本。

2. 软件需求

在 LNMP 架构中，操作系统自带的 RPM 版本都很低，性能和安全方面都存在一定的问題。为达到更好的性能和更高的安全需要，在大规模的生产环境中，选用最新版本的软件包成为必然。

对于软件包的安装问题，一是选用源码包，二是选用各软件包的官方发行 RPM 包，三是选用官方提供的二进制包，四是从源码自带的 Rpmbuild 构建形成 RPM 包。在这四种方式中，第四方式是比较推荐的做法，理由是 RPM 包容易管理，自己定制能满足个性化需求，例如，可以自定义各种软件包依赖关系，与前两种方式相比，Rpmbuild 花费的时间看似更多，但更利于后期维护和大规模部署。因此，这也是本书极力推荐的方式。RPMbuild 只适用于 RHEL、CentOS、SUSE 这几个 Linux 发行版本，如果是 Debian、Ubuntu 系统，则为 DEB 包管理方式，其原理大致相同。

读者可以通过下面的网址下载本章所用的 RPM 包（对于对应用需求较高的读者，可以用本书 github 中提供的 SRPM 格式 RPM 包对 Rpmbuild 进行打包并构建 RPM，对部分配置稍作修改即可满足需求）。

<https://github.com/itnihao/zabbix-book/tree/master/NMP>

16.3.2 Zabbix DB 的安装

Zabbix 的 DB 规划如表 16-3 所示。

表 16-3

角 色	IP 地址	域 名	运行的服务
MySQL-master	192.168.0.240	zabbix-mysql-master.itnihao.com	MySQL Server
MySQL-slave	192.168.0.241	zabbix-mysql-slave.itnihao.com	MySQL Server

分别在表 16-3 列出的机器中安装 MySQL 服务。

1. MySQL 的安装

在第 3 章中已经知道了，Zabbix 数据的存储成为监控系统的核心问题，故选择合适的数据库软件是非常重要的。这里选择 Percona 作为 Zabbix 的数据库软件，其理由是 Percona 对 MySQL 数据库进行了功能和性能方面的改进，尤其是对 InnoDB 进行了显著的性能提升。

MySQL 5.6 与 MySQL 5.5 相比，其性能差别非常大，因此，本环境的数据库选择了 Percona-Server-5.6，下载地址如下。

<http://www.percona.com/downloads/Percona-Server-5.6/LATEST/>

版本包括二进制、源码、RPM、deb 四种格式的包，如图 16-2 所示，具体选择哪种方式，读者可以自行考虑，这里考虑到方便快捷而选择了二进制的安装方式。

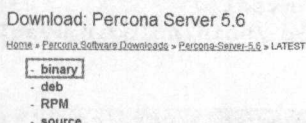


图 16-2

下载 Percona-Server 到本地，如图 16-3 所示，是一个二进制的包。

http://www.percona.com/downloads/Percona-Server-5.6/LATEST/binary/linux/x86_64/

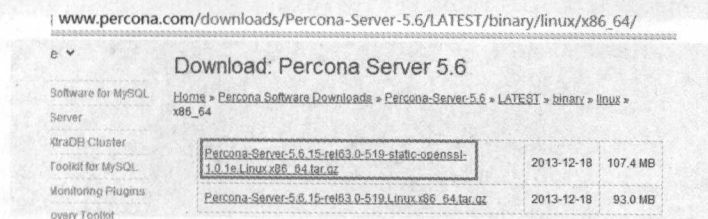


图 16-3

提示:官方的 RPM 包在 RHEL 6.3 系统中存在依赖关系而无法安装,如图 16-4 所示的依赖提示无法解决。

```
--> Running transaction check
--> Package Percona-Server-client-56.x86_64 0:5.6.15-rel63.0.519.rhel6 will be insta
--> Processing Dependency: libssl.so.10(libssl.so.10)(64bit) for package: Percona-Ser
--> Processing Dependency: libcrypto.so.10(libcrypto.so.10)(64bit) for package: Perco
--> Package Percona-Server-shared-56.x86_64 0:5.6.15-rel63.0.519.rhel6 will be obso
--> Processing Dependency: libssl.so.10(libssl.so.10)(64bit) for package: Percona-Ser
--> Processing Dependency: libcrypto.so.10(libcrypto.so.10)(64bit) for package: Perco
--> Package Percona-Server-shared-compat.x86_64 0:5.5.35-rel33.0.611.rhel6 will be d
--> Finished Dependency Resolution
Error: Package: Percona-Server-shared-56-5.6.15-rel63.0.519.rhel6.x86_64 (percona)
Requires: libcrypto.so.10(libcrypto.so.10)(64bit)
Error: Package: Percona-Server-shared-56-5.6.15-rel63.0.519.rhel6.x86_64 (percona)
Requires: libssl.so.10(libssl.so.10)(64bit)
Error: Package: Percona-Server-client-56-5.6.15-rel63.0.519.rhel6.x86_64 (percona)
Requires: libssl.so.10(libssl.so.10)(64bit)
Error: Package: Percona-Server-client-56-5.6.15-rel63.0.519.rhel6.x86_64 (percona)
Requires: libcrypto.so.10(libcrypto.so.10)(64bit)
```

图 16-4

对于以下内容,读者可参考 <https://github.com/itnihao/zabbix-book/blob/master/NMP/mysql-install-readme.md>, 这里以 Percona-Server-5.6.15-rel63.0-519 为例。

(1) 安装 Percona-Server

```
shell# wget http://www.percona.com/redir/downloads/Percona-Server
-5.6/LATEST/binary/linux/x86_64/Percona-Server-5.6.15-rel63.0-519-st
atic-openssl-1.0.1e.Linux.x86_64.tar.gz
shell# tar xf Percona-Server-5.6.15-rel63.0-519-static-openssl-
1.0.1e.Linux.x86_64.tar.gz
shell# mv Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1
e.Linux.x86_64 /usr/local/
shell# cd /usr/local/Percona-Server-5.6.15-rel63.0-519-static-op
enssl-1.0.1e.Linux.x86_64/
shell# cp support-files/mysql.server /etc/init.d/mysqld
```

(2) 建立用户

```
shell# groupadd -g 27 mysql
shell# useradd -g 27 -s /sbin/nologin mysql
```

(3) 改变权限

```
shell# chown -R mysql:mysql /usr/local/Percona-Server-5.6.15-rel
63.0-519-static-openssl-1.0.1e.Linux.x86_64/
```

提示:如果路径不为/usr/local,则需要修改启动脚本/etc/init.d/mysqld。

配置环境变量如图 16-5 所示。

```
shell# vim ~/.bash_profile
PATH=$PATH:$HOME/bin:/usr/local/Percona-Server-5.6.15-rel63.0-51
9-static-openssl-1.0.1e.Linux.x86_64/bin
```

```
[root@mysql ~]# vim ~/.bash_profile
# ~/.bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:/bin:/usr/local/Percona-Server-5.6.15-rel63.0-510-static-openssl-1.0.1e.Linux.x86_64/bin
export PATH
```

图 16-5

(4) 启动 Percona-Server 服务

注意文件/etc/my.cnf 配置的正确性, 否则, MySQL 不能正常启动, 因为 mysqld 脚本中的默认路径会读取/etc/my.cnf 文件的配置内容。

```
shell# mysqld --verbose --help|grep my.cnf
```

my.cnf 将会存在于以下路径, 依次为优先级匹配。

```
/etc/my.cnf
/etc/mysql/my.cnf /usr/local/Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64/etc/my.cnf
~/my.cnf
```

(5) 配置 my.cnf 文件

建立所需的目录, 存放配置文件、sock 和 pid 文件。

```
shell# mkdir /usr/local/Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64/etc
shell# mkdir /usr/local/Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64/var/run -p
shell# mkdir -p /usr/local/Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64/var/log
```

建立 my.cnf 文件, 语句如下:

```
shell# vim /usr/local/Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64/etc/my.cnf
[mysqld]
datadir=/usr/local/Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64/data
socket=/usr/local/Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64/var/run/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
character-set-server=utf8
innodb_file_per_table=1

[mysqld_safe]
log-error=/usr/local/Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64/var/log/mysqld.log
pid-file=/usr/local/Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64/var/run/mysqld.pid
```

以上的 my.cnf 为简单的参数配置, 后期还需要对此进行调整。

(6) 初始化 MySQL

```
shell# mkdir /opt/bak
shell# mv /etc/my.cnf /opt/bak
shell# chown -R mysql:mysql /usr/local/Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64/
shell# pwd
/usr/local/Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64/
```



```

shell# ./scripts/mysql_install_db \
--user=mysql \
--basedir=/usr/local/Percona-Server-5.6.15-rel63.0-519-static-op
enssl-1.0.1e.Linux.x86_64/ \
--datadir=/usr/local/Percona-Server-5.6.15-rel63.0-519-static-op
enssl-1.0.1e.Linux.x86_64/data/
shell# ./bin/mysqld_safe &
shell# cp support-files/mysql.server /etc/init.d/mysqld
shell# chkconfig mysqld on
shell# /etc/init.d/mysqld start

```

(7) 启动服务

启动 Percona-Server 服务，如图 16-6 所示。

```

[root@mysql Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64]# ps -ef | grep mysqld
[root@mysql Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64]# /etc/init.d/mysqld start
Starting MySQL (Percona Server) [ OK ]
[root@mysql Percona-Server-5.6.15-rel63.0-519-static-openssl-1.0.1e.Linux.x86_64]#

```

图 16-6

登录 MySQL，如图 16-7 所示。

```

[root@mysql ~]# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.15-rel63.0 Percona Server with XtraDB (GPL), Release rel63.0, Revision 519
Copyright (c) 2009-2013 Percona LLC and/or its affiliates
Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

```

图 16-7

说明 MySQL 安装配置成功，若无法连接到 sock 文件，则解决方法如下：

```

shell# mysql -p
Enter password:
ERROR 2002 (HY000): Can't connect to local MySQL server through so
cket '/tmp/mysql.sock' (2)
shell# mysql -p --socket=/usr/local/Percona-Server-5.6.15-
rel63.0-519-static-openssl-1.0.1e.Linux.x86_64/var/run/mysql.sock

```

2. 创建 Zabbix 数据库

```

shell# MysqlPassword=admin
shell# mysqladmin -u root password ${MysqlPassword}
mysql> create database zabbix character set utf8;
mysql> grant all privileges on zabbix.* to zabbix@'192.168.0.2' id
entified by 'zabbix';
mysql> grant all privileges on zabbix.* to zabbix@'192.168.0.3' id
entified by 'zabbix';
mysql> grant all privileges on zabbix.* to zabbix@'192.168.0.4' id
entified by 'zabbix';

```

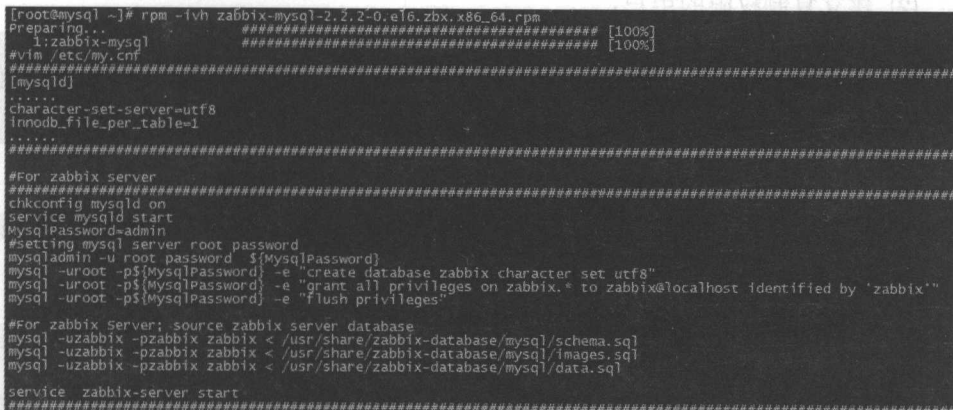


```
mysql> grant all privileges on zabbix.* to zabbix@'192.168.0.5' id
entified by 'zabbix';
mysql> flush privileges;
```

3. 安装 Zabbix-MySQL

```
shell# rpm -ivh zabbix-mysql-2.2.2-0.el6.zbx.x86_64.rpm
```

安装过程如图 16-8 所示。



```
[root@mysql ~]# rpm -ivh zabbix-mysql-2.2.2-0.el6.zbx.x86_64.rpm
Preparing...
1:zabbix-mysql
#vim /etc/my.cnf
[mysql]
.....
character-set-server=utf8
innodb_file_per_table=1
.....
#For zabbix server
#####
chkconfig mysqld on
service mysqld start
MySQLPassword=admin
#setting mysql server root password
mysqladmin -u root password ${MySQLPassword}
mysql -uroot -p${MySQLPassword} -e "create database zabbix character set utf8"
mysql -uroot -p${MySQLPassword} -e "grant all privileges on zabbix.* to zabbix@localhost identified by 'zabbix'"
mysql -uroot -p${MySQLPassword} -e "flush privileges"
#For zabbix Server: source zabbix server database
mysql -uzabbix -pzabbix zabbix < /usr/share/zabbix-database/mysql/schema.sql
mysql -uzabbix -pzabbix zabbix < /usr/share/zabbix-database/mysql/images.sql
mysql -uzabbix -pzabbix zabbix < /usr/share/zabbix-database/mysql/data.sql
service zabbix-server start
#####
```

图 16-8

4. 导入 Zabbix-Server 的数据库

```
shell# mysql -uzabbix -pzabbix zabbix < /usr/share/zabbix-datab
ase/mysql/schema.sql
#zabbix_proxy只需导入schema.sql, 下面的两个无须导入
shell# mysql -uzabbix -pzabbix zabbix < /usr/share/zabbix-datab
ase/mysql/images.sql
shell# mysql -uzabbix -pzabbix zabbix < /usr/share/zabbix-datab
ase/mysql/data.sql
```

5. MySQL 主从的配置

(1) 在 MySQL 主服务器 192.168.0.240 上配置

① 修改 my.cnf 文件。

```
shell# vim /usr/local/Percona-Server-5.6.15-rel63.0-519-static-op
enssl-1.0.1e.Linux.x86_64/etc/my.cnf
[mysql]
.....
server-id = 1
log-bin=mysql-bin
log-bin=/usr/local/Percona-Server-5.6.15-rel63.0-519-static-open
ssl-1.0.1e.Linux.x86_64/var/log/update
mysql> show master status;
+-----+-----+-----+-----+-----+-----+
--+
```

File	Position	Binlog_Do_DB	Binlog_Ignore_DB	Executed_Gtid_Set
update.000001	107			

② 建立复制权限的用户。

```
shell# mysql -uroot -p
mysql> GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO repl@'192.168.0.241.%' IDENTIFIED BY 'zabbix_repl';
mysql> FLUSH PRIVILEGES;
```

③ 备份数据库。

```
shell# mysqldump -uzabbix -pzabbix >zabbix.sql
```

注意：备份完成后复制到从库，并停止对主库的写操作。

(2) 在 MySQL 从服务器 192.168.0.241 上配置

① 修改 my.cnf 文件。

```
[mysqld]
.....
server-id = 2 #1表示master、slave依次增大
log-bin=mysql-bin
```

② 导入 Master 中的数据库。

```
shell# mysql -uzabbix -pzabbix <zabbix.sql
```

③ 配置复制权限。

```
mysql> change master to master_host='192.168.0.240', MASTER_USER='repl', MASTER_PASSWORD='zabbix_repl', MASTER_PORT=3306, MASTER_LOG_FILE='update.000001', MASTER_LOG_POS=107, MASTER_CONNECT_RETRY=10;
#107要和mater的Position数值一致
```

④ 启动 MySQL slave。

```
mysql> start slave;
mysql> show slave status\G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.0.240
Master_User: repl
Master_Port: 3306
Connect_Retry: 10
Master_Log_File: updatelog.000007
Read_Master_Log_Pos: 888190
Relay_Log_File: mysql-relay-bin.000002
Relay_Log_Pos: 107
Relay_Master_Log_File: updatelog.000007
```

16.3.3 安装 Zabbix-Server

Zabbix-Server 的规划如表 16-4 所示。

表 16-4

角 色	IP 地址	域 名	运行的服务
zabbix-master	192.168.0.3	zabbix-server-master.itnihao.com	zabbix server
zabbix-slave	192.168.0.4	Zabbix-server-slave.itnihao.com	Zabbix server

分别在表 16-4 列出的机器中安装 Zabbix-Server 服务。
安装 Zabbix-Server，如图 16-9 所示。

```
Running Transaction
Installing : libdbi-0.8.3-4.el6.x86_64
Installing : libdbi-drivers-0.8.3-5.1.el6.x86_64
Installing : libdbi-dbd-mysql-0.8.3-5.1.el6.x86_64
Installing : zabbix-server-2.2.2-0.el6.zbx.x86_64
```

图 16-9

```
shell# egrep -v "^\$|^#" /etc/zabbix/zabbix_server.conf
LogFile=/var/log/zabbix/zabbix_server.log
LogFileSize=0
PidFile=/var/run/zabbix/zabbix_server.pid
DBHost = zabbix-mysql-master.itnihao.com
DBName=zabbix
DBUser=zabbix
DBPassword=zabbix
DBSocket=/var/lib/mysql/mysql.sock
SNMPTrapperFile=/var/log/snmpd/snmpd.log
AlertScriptsPath=/etc/zabbix/alertscripts
ExternalScripts=/etc/zabbix/externalscripts
```

注意查看日志信息，提示数据库是否连接正常，如图 16-10 所示。

```
[root@zabbix-master zabbix]# tail -f /var/log/zabbix/zabbix_server.log
29032:20140223:231054.435 server #0 started [main process]
29116:20140223:231054.435 server #26 started [self-monitoring #1]
29106:20140223:231055.256 housekeeper [deleted 0 hist:trends, 0 items, 0
29093:20140223:231055.502 server #5 started [poller #3]
29093:20140223:231055.505 server #3 started [poller #1]
29096:20140223:231055.505 server #6 started [poller #4]
29109:20140223:231055.507 server #19 started [discoverer #1]
29097:20140223:231055.509 server #7 started [poller #5]
29098:20140223:231055.512 server #8 started [unreachable poller #1]
29094:20140223:231055.516 server #4 started [poller #2]
```

图 16-10

Zabbix-Server 的 HA 可以用很多 HA 软件来实现，这类软件包括 Corosync + Pacemaker、keepalived 或 RHCS 等。这里选用最简单的 keepalived+脚本方式来实现 Zabbix-Server 高可用的配置。

由于 Agent 连接 Zabbix-Server 的进程，在这个时候，Zabbix-Server 出现了故障，如果数据没有及时写入 Zabbix 数据库中，会造成监控数据在这段时间丢失，因此，可能会出现短暂的误报（通过配置合理的 Trigger 即可解决）。

表 16-5 是本环境中各服务器的用途。

表 16-5

角 色	IP 地址	虚拟 IP	运行的服务
Zabbix-Server-Master	192.168.0.3	192.168.0.5	Zabbix-server Keepalived(Master)
Zabbix-Server-Slave	192.168.0.4		Zabbix-server Keepalived()

1. 安装 Keepalived

在 RHEL、CentOS 6.4 以上的版本中有 RPM 包的 Keepalived，可以直接下载安装。

```
shell# rpm -ivh http://mirrors.sohu.com/centos/6.5/os/x86_64/Packages/keepalived-1.2.7-3.el6.x86_64.rpm
```

2. 配置 Keepalived

```
shell#cat /etc/keepalived/keepalived.conf
global_defs {
    notification_email {
        admin@itnihao.com
    }
    notification_email_from zabbix-master@itnihao.com
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
}

vrrp_script chk_zabbix_server {
    script "/etc/keepalived/chk_zabbix_server.sh"
    interval 30          #检测间隔时间为30s
    weight 2
}

vrrp_instance VI_1 {
    state MASTER #192.168.0.4中配置为BACKUP
    interface eth0
    virtual_router_id 51
    priority 111 #
    advert_int 1
    mcast_src_ip 192.168.0.3

    authentication {
        auth_type PASS
        auth_pass ZabbixMonitor
    }
}
```



```
track_script {
    chk_zabbix_server
}

#VIP
virtual_ipaddress {
    192.168.0.5
}
}
```

3. Keepalived 检测脚本

```
shell#cat /etc/keepalived/chk_zabbix_server.sh
#!/bin/bash
#
#
status1=$(ps aux|grep "/usr/sbin/zabbix_server" | grep -v grep | g
rep -v bash | wc -l)

if [ "${status1}" = "0" ]; then

    /etc/init.d/zabbix-server start
    sleep 3

    status2=$(ps aux|grep zabbix_server | grep -v grep | grep -
v bash |wc -l)
    if [ "${status2}" = "0" ]; then
        /etc/init.d/keepalived stop
    fi
fi
```

16.3.4 安装 Zabbix-GUI

Zabbix-GUI 的规划如表 16-6 所示。

表 16-6

角 色	IP 地址	域 名	运行的服务
Zabbix-gui	192.168.0.2	Zabbix-gui.itnihao.com	Php-fpm nginx

1. Nginx 的安装

由于这里的 Zabbix 需要对外提供 API 数据，所以 Web 服务器需要一个性能更高的选择。在我们的使用经验中，Nginx 在静态处理能力方面明显比 APACHE 要强很多。Nginx 在和 PHP 的组合中，可以通过多种方式进行扩展，从而提高 PHP 的处理能力，例如，PHP 开启多个进程的方法。

Nginx 采用 RPM 包安装，如果是源码安装，请参考相关文档。

```
http://nginx.org/packages/rhel/6/x86_64/RPMS/
```

请读者下载 Nginx 最新的 RPM 进行安装, 如图 16-11 所示为安装 Nginx 的语句。

```
[root@zabbix-master nginx]# wget http://nginx.org/packages/rhel/6/x86_64/RPMS/nginx-1.4.5-1.el6ngx.x86_64.rpm
--2014-02-22 20:43:19-- http://nginx.org/packages/rhel/6/x86_64/RPMS/nginx-1.4.5-1.el6ngx.x86_64.rpm
Resolving nginx.org... 206.251.255.63
Connecting to nginx.org|206.251.255.63|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 318280 (311K) [application/x-redhat-package-manager]
Saving to: "nginx-1.4.5-1.el6ngx.x86_64.rpm"

100%[=====]
2014-02-22 20:43:22 (115 KB/s) - "nginx-1.4.5-1.el6ngx.x86_64.rpm" saved [318280/318280]

[root@zabbix-master nginx]# ls
nginx-1.4.5-1.el6ngx.x86_64.rpm
[root@zabbix-master nginx]# rpm -lvh nginx-1.4.5-1.el6ngx.x86_64.rpm
warning: nginx-1.4.5-1.el6ngx.x86_64.rpm: Header V4 RSA/SHA1 Signature, key ID 7bd9bf62: NOKEY
Preparing... [100%]
1:nginx [100%]

Thanks for using nginx!

Please find the official documentation for nginx here:
* http://nginx.org/en/docs/

Commercial subscriptions for nginx are available on:
* http://nginx.com/products/
```

图 16-11

Nginx 的配置如下:

```
cat /etc/nginx/conf.d/zabbix-nginx-web.conf
server {
    listen 80;
    server_name localhost;
    charset utf-8;
    index index.htm index.html index.php;
    access_log /var/log/nginx/zabbix.access.log main;
    error_log /var/log/nginx/zabbix.error.log;
    root /usr/share/zabbix;

    location / {
        index index.html index.htm index.php;
    }

    location ~ .*\. (php|php5)? $
    {
        #fastcgi_pass 127.0.0.1:9000;
        fastcgi_pass unix:/var/run/php/php-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastc
gi_script_name;
        include fastcgi_params;
    }
}
shell#service nginx restart
```

2. PHP-FPM 的安装

这里以 RPM 为例, 介绍 PHP-FPM 的安装。如果是其他平台, 读者可以通过

源码方式进行 PHP-FPM 的安装。

PHP 所需的文件在 <https://github.com/itnihao/zabbix-book/tree/master/NMP> 中下载，需要解决如图 16-12 所示的依赖包。

```
Installing : libmcrypt-2.5.8-9.el6.x86_64
Installing : libicu-4.2.1-9.1.el6_2.x86_64
Installing : libtidy-0.99.0-19.20070615.1.el6.x86_64
Installing : libjpeg-turbo-1.2.1-1.el6.x86_64
Installing : php-5.4.25-1.el6.x86_64
```

图 16-12

PHP-FPM 的配置如下：

```
shell# vim /etc/php.ini
date.timezone = Asia/Shanghai
max_execution_time = 300
post_max_size = 16M
max_input_time=300
memory_limit = 128M
mbstring.func_overload = 2
shell# service php-fpm restart
```

用 RPM 包安装，如图 16-13 所示。

```
[root@zabbix-gui ~]# rpm -ivh zabbix-web-nginx-2.2.2-0.el6.zbx.noarch.rpm
Preparing... [100%]
1:zabbix-web-nginx [100%]
/var/tmp/rpm-tmp.81CTgp: line 4: semanage: command not found
chown: invalid user: www.www
mkdir: cannot create directory '/etc/nginx/conf.d/bak': File exists
Gracefully shutting down php-fpm . done
Starting php-fpm done
Stopping nginx: [ OK ]
Starting nginx: [ OK ]
[root@zabbix-gui ~]#
```

图 16-13

下面进行 Web 页面的配置，访问地址如下：

<http://zabbix-gui.itnihao.com/inex.php> #界面如图16-14所示

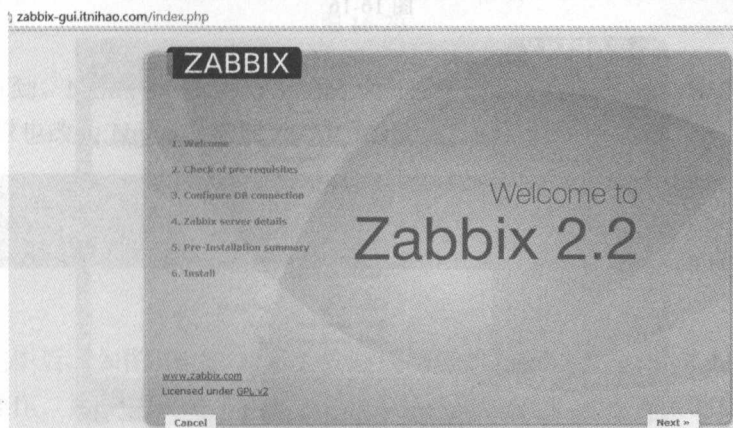


图 16-14

如果出现数据库无法连接，则需正确配置数据库，如图 16-15 所示。

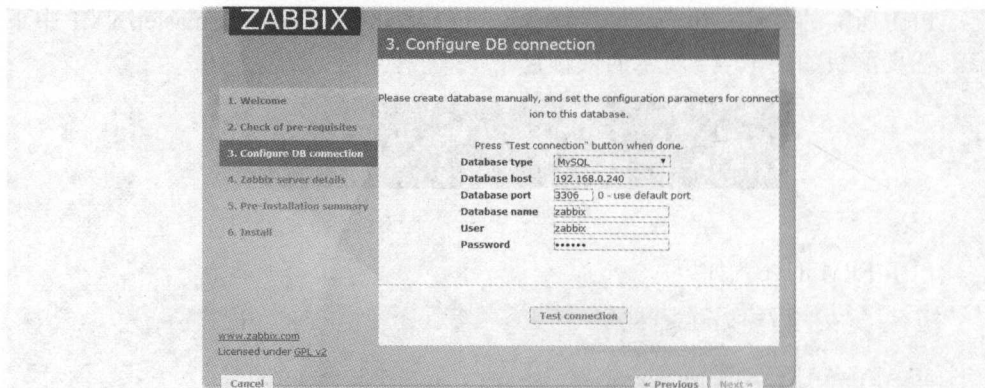


图 16-15

填写虚拟的 IP 地址：192.168.0.5，如图 16-16 所示，图 16-17 显示了当前的配置环境信息。

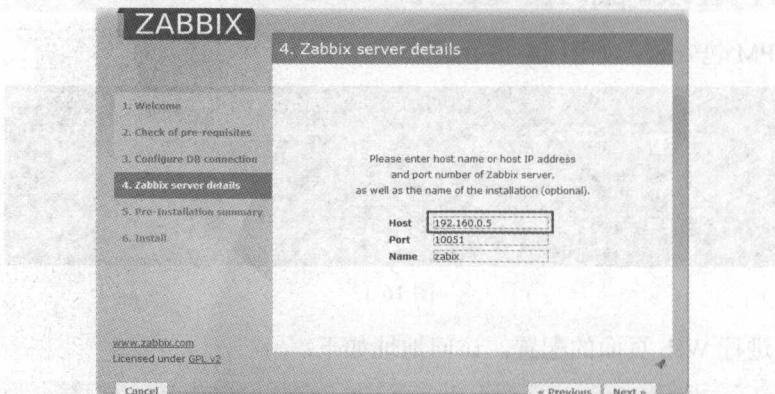


图 16-16

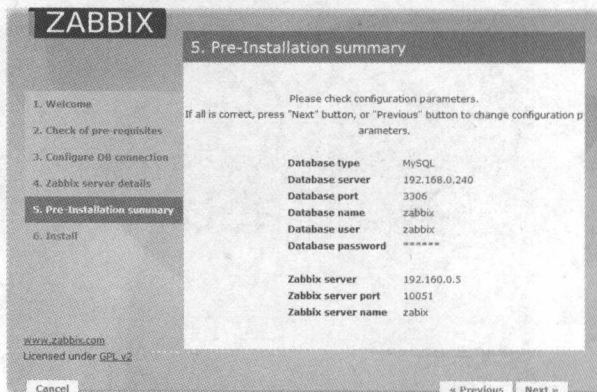


图 16-17

提示无法写入，如图 16-18 所示。

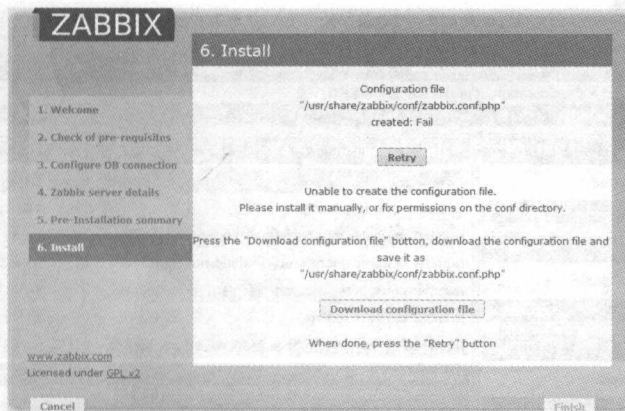


图 16-18

查看文件夹权限，如图 16-19 所示。

```

-rw-r--r--. 1 nginx nginx 427 Feb 12 17:04 zabbix.conf.php.example
[root@zabbix-gui ~]# ll /usr/share/zabbix/conf/
total 8
-rw-r--r--. 1 nginx nginx 1036 Feb 12 17:04 maintenance.inc.php
-rw-r--r--. 1 nginx nginx 427 Feb 12 17:04 zabbix.conf.php.example

```

图 16-19

查看 Nginx 和 PHP 的权限，如图 16-20 所示。

```

[root@zabbix-gui ~]# ps -ef |grep php
root      5177      1    0 23:16 ?        00:00:00 php-fpm: master process (/etc/php-fpm.conf)
nobody    5178    5177    0 23:16 ?        00:00:02 php-fpm: pool www
nobody    5179    5177    0 23:16 ?        00:00:02 php-fpm: pool www
root      5346    4235    0 23:30 pts/1    00:00:00 grep php

[root@zabbix-gui ~]# ps -ef |grep nginx
root      5233      1    0 23:18 ?        00:00:00 nginx: master process /usr/sbin/nginx -c /etc/
nginx     5234    5233    0 23:18 ?        00:00:01 nginx: worker process
root      5348    4235    0 23:31 pts/1    00:00:00 grep nginx

```

图 16-20

可以看到，Nginx 和 PHP 运行在两个用户中，所以无法写入，解决的办法是将 PHP 用户也改为 Nginx，如图 16-21 所示。

```

: unix user/group of processes
: Note: The user is mandatory. If the group is not set, the default user's group
: will be used.
user = nginx
group = nginx

```

图 16-21

安装完毕后（如图 16-22 所示为安装成功后的界面），需要对 Zabbix 数据库进行分表操作。第 3 章已经讲解过了，此处不重复。有关配置参数的优化、模板的优化、触发器的优化等内容，在前面章节已经讲解。

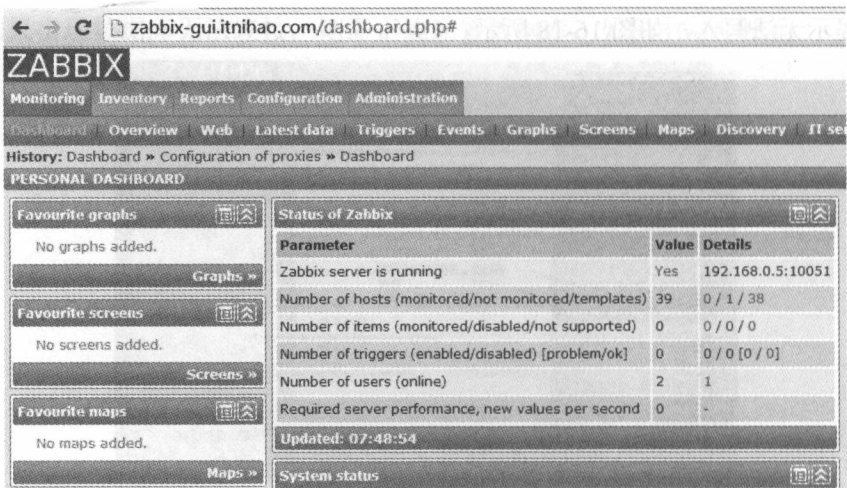


图 16-22

16.3.5 安装 Zabbix-Proxy

Zabbix-Proxy 的规划如表 16-7 所示。

表 16-7

角 色	IP 地址	域 名	运行的服务
Zabbix-Proxy	192.168.0.6	zabbix-proxy.itnihao.com	Zabbix-proxy MySQL Server

安装 Zabbix-Proxy，语句如下。

```
shell# rpm -ivh zabbix-proxy-2.2.2-0.el6.zbx.x86_64.rpm
```

安装 Zabbix-MySQL，语句如下。

```
shell# rpm -ivh zabbix-mysql-2.2.2-0.el6.zbx.x86_64.rpm
```

安装 Percona 数据库软件。

安装步骤请参考 16.3.2 节的内容。

创建 Zabbix-Proxy 数据库，如图 16-23 所示。

```
mysql> create database zabbix_proxy character set utf8;
Query OK, 1 row affected (0.15 sec)

mysql> grant all privileges on zabbix_proxy.* to zabbix@localhost identified by 'zabbix';
Query OK, 0 rows affected (0.15 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.05 sec)

mysql> use zabbix_proxy;
Database changed
mysql> source /usr/share/zabbix-database/mysql/schema.sql;
```

图 16-23

修改 zabbix_proxy.conf 配置。

```
shell# egrep -v "^\$|^#" /etc/zabbix/zabbix_proxy.conf
ProxyMode=0
Server=192.168.0.5
Hostname=Zabbix proxy
LogFile=/var/log/zabbix/zabbix_proxy.log
LogFileSize=0
PidFile=/var/run/zabbix/zabbix_proxy.pid
DBName=zabbix_proxy
DBUser=zabbix
DBPassword=zabbix
DBSocket=/usr/local/Percona-Server-5.6.15-rel63.0-519-static-open
ssl-1.0.1e.Linux.x86_64/var/run/mysql.sock
ExternalScripts=/etc/zabbix/externalscripts
```

启动 Zabbix-Proxy 服务。

```
shell# /etc/init.d/zabbix-proxy start
```

检查日志，查看启动是否成功，是否有报错等，如图 16-24 所示。

```
[root@zabbix-proxy ~]# tail -f /var/log/zabbix/zabbix_proxy.log
9140:20140225:155950.594 housekeeper [deleted 0 records in 0.020976 sec, idle 3600 sec]
9121:20140225:155950.823 sending heartbeat message to server failed: error:"negative response:
9120:20140225:155950.823 Cannot obtain configuration data from server. info:"proxy "zabbix proxy
9125:20140225:155951.253 proxy #6 started [poller #3]
9123:20140225:155951.254 proxy #4 started [poller #1]
9127:20140225:155951.255 proxy #8 started [poller #5]
9124:20140225:155951.255 proxy #5 started [poller #2]
9128:20140225:155951.257 proxy #9 started [unreachable poller #1]
9143:20140225:155951.266 proxy #18 started [discoverer #1]
9126:20140225:155951.268 proxy #7 started [poller #4]
```

图 16-24

配置 Zabbix-Proxy，如图 16-25 所示。单击 Administration→DM→Proxies→Create proxy。

The screenshot shows the Zabbix web interface for configuring a proxy. The 'Administration' tab is selected, and the 'Proxies' section is active. The 'Create proxy' form is displayed with the following details:

- Proxy name:** Zabbix proxy
- Proxy mode:** Active
- Hosts:** (Empty list)
- Proxy hosts:** (Empty list)
- Other hosts:** Zabbix server

At the bottom of the form, there are buttons for 'Save', 'Clone', 'Delete', and 'Cancel'.

图 16-25

注意，图 16-25 中的 Poryx name 来自 zabbix_proxy.conf 下的 Hostname，每个 Zabbix-Proxy 应该具有唯一的名称，不可重复，如图 16-26 所示。

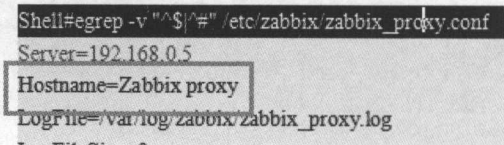


图 16-26

添加自动注册的 Actions，如图 16-27 到图 16-29 所示。

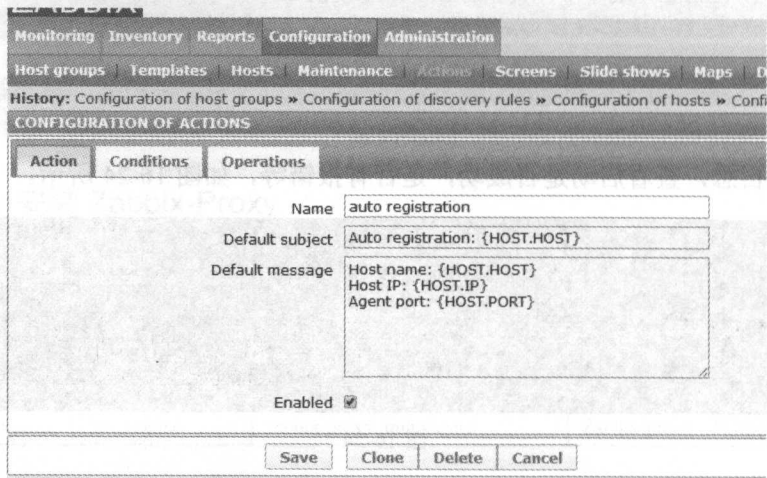


图 16-27

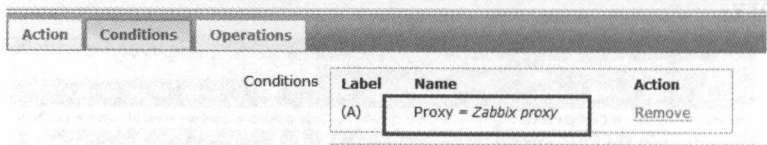


图 16-28

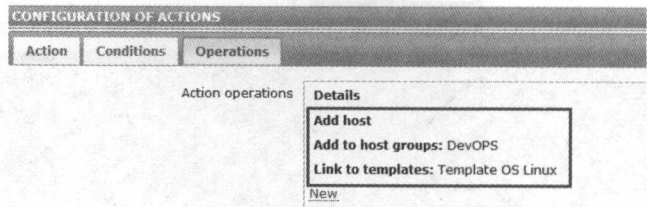


图 16-29

添加完成后的结果如图 16-30 所示。

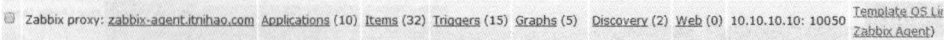


图 16-30

16.3.6 配置 Zabbix-Agent

修改 /etc/zabbix/zabbix_agentd.conf，语句如下。

```
ServerActive=127.0.0.1:10051,10.10.10.2:10051
```

注意：10.10.10.2 为 Zabbix-Proxy 的内网 IP 地址。

16.4 业务相关的配置

16.4.1 用户的配置

通过单击 Administration→Users→Users groups→Create user group，创建需要的业务组用户，如图 16-31 所示。

ZABBIX					
Help Get support Print Profile Logout					
Monitoring Inventory Reports Configuration Administration					
General DM Authentication Users Media types Scripts Audit Queue Notifications Installation					
History: Configuration of users » Configuration of user groups » Configuration of users » Configuration of user groups » Configuration of users					
CONFIGURATION OF USER GROUPS					
User groups					
Displaying 1 to 19 of 19 found					
Name	#	Members	Status	Frontend access	Debug mode
Boss	Users (1)	boss (boss wu Mr wu)	Enabled	System default	Disabled
business manager group1	Users (1)	user1 (user1 user1)	Enabled	System default	Disabled
business manager group2	Users (1)	user2 (user2 user2)	Enabled	System default	Disabled
business manager group3	Users (1)	user3 (user3 user3)	Enabled	System default	Disabled
DevOps	Users (1)	Devops Li (Li ming Mr Li)	Enabled	System default	Disabled
Dev web app group1	Users (1)	web1 Song (Song wei Mr Song)	Enabled	System default	Disabled
Dev web app group2	Users (1)	web2 Zhao (Zhao ming Mr Zhao)	Enabled	System default	Disabled
Dev web app group3	Users (1)	web3 Tang (Tang zong Mr Tang)	Enabled	System default	Disabled
Disabled	Users (0)		Disabled	System default	Disabled
Enabled debug mode	Users (0)		Enabled	System default	Enabled
Guests	Users (1)	guest	Enabled	System default	Disabled
Network OPS	Users (1)	Network Hu (Hu wei Mr Hu)	Enabled	System default	Disabled
No access to the frontend	Users (0)		Enabled	Disabled	Disabled
Security	Users (0)		Enabled	System default	Disabled
SYS OPS group1	Users (0)		Enabled	System default	Disabled
SYS OPS group2	Users (0)		Enabled	System default	Disabled

图 16-31

通过单击 Administration→Users→Users→Create user，依次创建需要的用户，如图 16-32 所示。

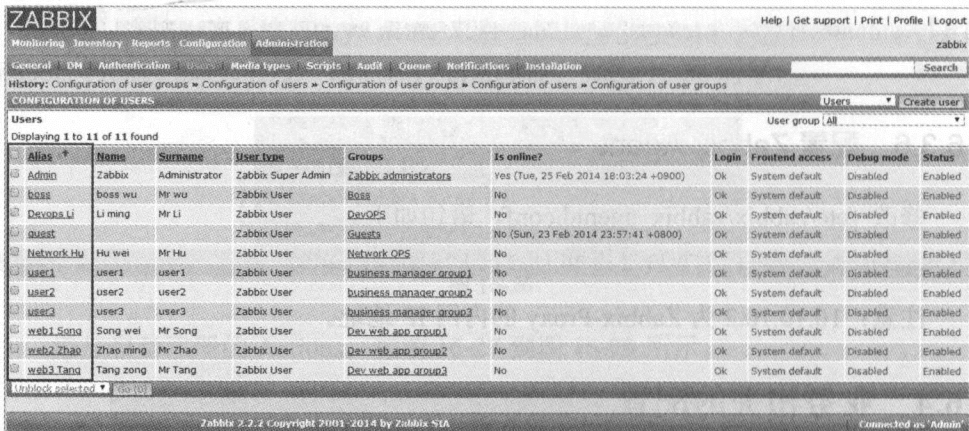


图 16-32

16.4.2 业务组的配置

业务的分组规则见 4.2 节介绍的原则，如图 16-33 所示，是划分了多个业务组的界面。

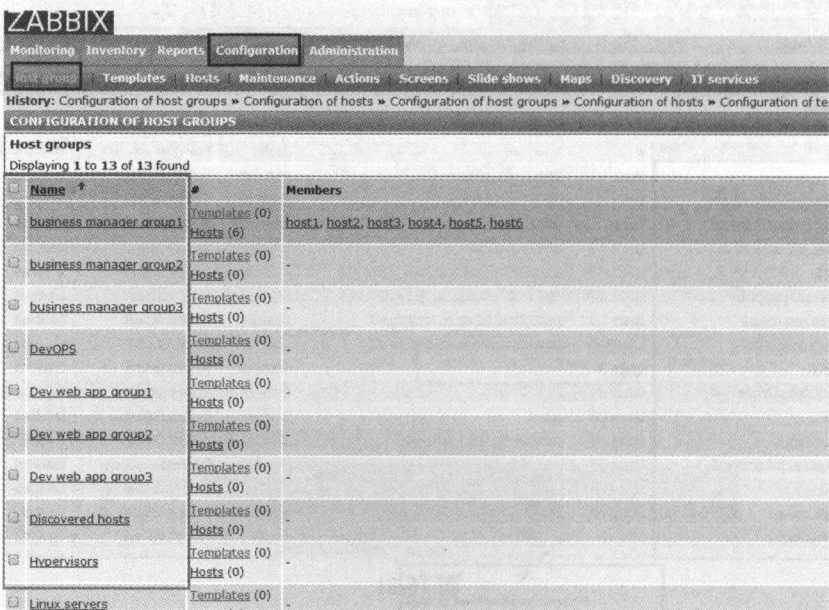


图 16-33

16.4.3 监控模板的定制

如图 16-34 所示的模板是 Zabbix 提供的。

Monitoring Inventory Reports Configuration Administration								
Host groups Hosts Maintenance Actions Screens Slide shows Maps Discovery IT-services								
History: Configuration of hosts » Configuration of host groups » Configuration of hosts » Configuration of templates » Configuration of host groups								
CONFIGURATION OF TEMPLATES								
Templates								
Displaying 1 to 38 of 38 found								
Templates	Applications	Items	Triggers	Graphs	Screens	Discovery	Web	Linked templates
Template App FTP Service	Applications (1)	Items (1)	Triggers (1)	Graphs (0)	Screens (0)	Discovery (0)	Web (0)	-
Template App HTTP Service	Applications (1)	Items (1)	Triggers (1)	Graphs (0)	Screens (0)	Discovery (0)	Web (0)	-
Template App HTTPS Service	Applications (1)	Items (1)	Triggers (1)	Graphs (0)	Screens (0)	Discovery (0)	Web (0)	-
Template App IMAP Service	Applications (1)	Items (1)	Triggers (1)	Graphs (0)	Screens (0)	Discovery (0)	Web (0)	-
Template App LDAP Service	Applications (1)	Items (1)	Triggers (1)	Graphs (0)	Screens (0)	Discovery (0)	Web (0)	-
Template App MySQL	Applications (1)	Items (14)	Triggers (1)	Graphs (2)	Screens (1)	Discovery (0)	Web (0)	-
Template App NNTP Service	Applications (1)	Items (1)	Triggers (1)	Graphs (0)	Screens (0)	Discovery (0)	Web (0)	-
Template App NTP Service	Applications (1)	Items (1)	Triggers (1)	Graphs (0)	Screens (0)	Discovery (0)	Web (0)	-
Template App POP Service	Applications (1)	Items (1)	Triggers (1)	Graphs (0)	Screens (0)	Discovery (0)	Web (0)	-
Template App SMTP Service	Applications (1)	Items (1)	Triggers (1)	Graphs (0)	Screens (0)	Discovery (0)	Web (0)	-
Template App SSH Service	Applications (1)	Items (1)	Triggers (1)	Graphs (0)	Screens (0)	Discovery (0)	Web (0)	-
Template App Telnet Service	Applications (1)	Items (1)	Triggers (1)	Graphs (0)	Screens (0)	Discovery (0)	Web (0)	-
Template App Zabbix Agent	Applications (1)	Items (3)	Triggers (3)	Graphs (0)	Screens (0)	Discovery (0)	Web (0)	-

图 16-34

在默认模板中，Items 工作在被动方式中。在实际环境中，由于监控的机器数量较多，所以需要配置为主动模式。具体配置主动模式的内容请参考 8.3 节。

其他需要优化的地方，如 Items 中历史数据的存储时间设置，将 History 和 Trends 的保存时间设置为更短，可以有效地减轻数据库的压力。

默认的触发器中，有很多是不合理的，例如，Template OS Linux 模板中，以下触发器的设置都不合理，需要根据实际情况来修改。

- {Template OS Linux:system.cpu.load[percpu,avg1].avg(5m)}>5: 单颗 CPU 负载
- {Template OS Linux:proc.num[.].avg(5m)}>300: 进程数量。
- {Template OS Linux:agent.ping.nodata(5m)}=1: 由于网络抖动而引起的误报。
- {Template OS Linux:kernel.maxfiles.last(0)}<1024: 文件描述符，实际大于此参数。
- {Template OS Linux:kernel.maxproc.last(0)}<256: 进程数，实际大于此参数。

在 Discovery 设置中，默认时间为 3600s，如图 16-35 所示，即一小时运行一次，对于初次构建的监控系统，可以将此值设置短一些，以便快速发现监控项，在所有的机器已经自动发现完毕后，在模板中将自动发现功能关闭，这样能减轻服务器的负担。

◀ Template list Template: Template OS Linux ▶ Discovery list Discovery: Mounted filesystem

Discovery rule

Name

Mounted filesystem discovery

Type

Zabbix agent

Key

vfs.fs.discovery

Update interval (in sec)

3600

Flexible intervals

Interval Period Action

No flexible intervals defined.

图 16-35

同时，由于我们需要的 Items 不存在，故需要添加新的 Items，添加包括自定义的 key 和 Zabbix 内置的 key。另外，模板中自带的 key 若对我们的业务无用，则需要将 Items 删除。关于 Items 的添加，请读者参考 5.2 节。

另一方面，我们根据业务进行模板的定制，例如，有的业务负载为 0.5 就告警，而有的业务负载为 10 却正常，内存、磁盘空间的利用率在不同的业务中也不相同，故需要我们对触发器按业务分类。

16.4.4 自动发现的配置

有关自动发现的内容，请读者参考 8.2 节。

在 Action 中，有关不同的机器组、不同业务的机器、添加不同的模板和不同分组的知识，前面已有介绍，此处不再重复。

16.5 其他需求

更多关于与其他资产系统结合的内容，比如 CMDB 系统、OPS 系统、运营系统等，需要进行少量的开发方可完成，如图 16-36 所示，为一个自动化运维平台的架构图。

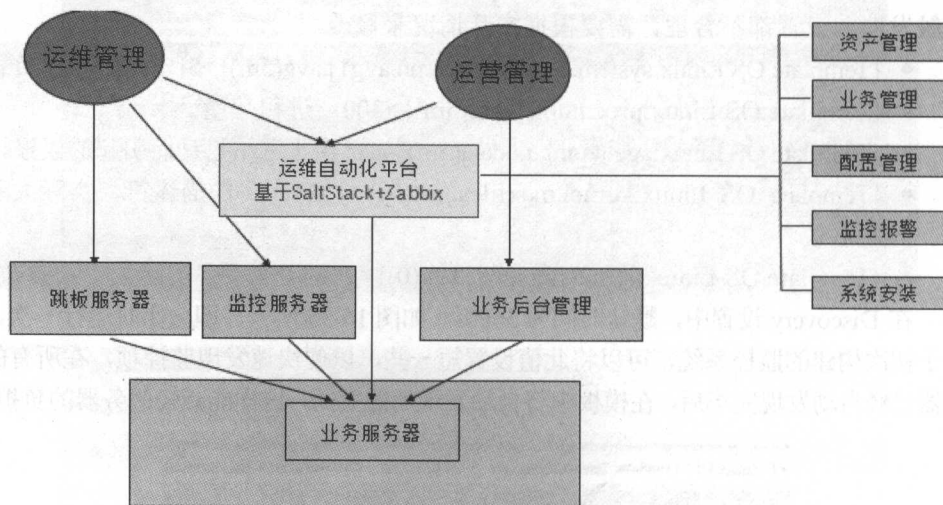


图 16-36

附录 A 源码安装及相关配置

考虑到其他的 Linux 平台不能通过 RPM 方式安装, 故这里介绍源码安装的示例, 但笔者并不推荐在大规模的批量部署环境中使用源码安装。

A.1 安装 Zabbix-Server

安装平台为 RHEL 6.X 或者 CentOS 6.X 操作系统。

1. 安装依赖包

```
shell# yum -y install gcc gcc-c++ autoconf httpd php mysql mysql
-server php-mysql httpd-manual mod_ssl mod_perl mod_auth_mysql php-gd
php-xml php-mbstring php-ldap php-pear php-xmlrpc php-bcmath mysql-c
onnector-odbc mysql-devel libdbi-dbd-mysql net-snmp-devel curl-devel
unixODBC-devel OpenIPMI-devel java-devel
```

如果是其他平台, 则软件包名字可能会有差别。

2. 配置 PHP 所需环境

```
shell# vim /etc/php.ini
date.timezone = Asia/Shanghai
max_execution_time = 300
post_max_size = 32M
max_input_time=300
memory_limit = 128M
mbstring.func_overload = 2
```

3. 安装 Zabbix-Server

```
shell# wget http://sourceforge.net/projects/zabbix/files/ZABBIX%
20Latest%20Stable/2.2.2/zabbix-2.2.2.tar.gz
```

(读者可以下载最新版本, 在写本书时的最新版本为 Zabbix 2.2.2, 无论是哪个版本, 安装差别不大。)

增加 zabbix 用户

```
shell# groupadd zabbix -g 201
shell# useradd -g zabbix -u 201 -m zabbix
shell# tar xvf zabbix-X.X.X.tar.gz
shell# cd ${zabbix-X.X.X }
```

注意, 以下用 \${zabbix-X.X.X} 代替 zabbix 的源码包所在的路径

```
shell# ./configure --prefix=/usr --sysconfdir=/etc/zabbix --ena
ble-server --enable-proxy --enable-agent --enable-ipv6 --with-mysq
l=/usr/bin/mysql_config --with-net-snmp --with-libcurl --with-openi
pmi --with-unixodbc --with-ldap --with-ssh2 --enable-java
```


(如果只想安装一个服务器端, 只需开启--enable-server 即可, 其他参数可不用选。这里是为了后面的各项功能都可以使用, 所以开启了非常多的参数。)

(如果缺少相应的依赖包, configure过程会给出提示, 用yum安装缺少的软件即可顺利通过安装。)

```
shell# make
shell# make install
```

4. 导入数据库

```
shell# cd ${zabbix-X.X.X} #确保路径在Zabbix源码下
shell# chkconfig mysql-server on
shell# service mysql-server start
shell# mysqladmin -uroot password 'mysql_pass';
(设置MySQL的root密码为mysql_pass。)
shell# mysql -uroot -p (登录数据库, 需输入刚才设置的密码。)
mysql> create database zabbix character set utf8;
(此处要特别注意数据库字符集的问题, 如果数据库是非utf8字符, 则Web页面改为中文后会出现乱码。)
mysql> grant all privileges on zabbix.* to zabbix@localhost
identified by 'zabbix';
mysql> flush privileges;
确保以上操作都正常, 测试数据库连接是否正常。
shell# mysql -uzabbix -pzabbix zabbix
如果可以正常连接, 说明用户名和密码正确, 连接正常后退出, 继续进行下面的操作。
shell# mysql -uzabbix -pzabbix zabbix < ${zabbix-X.X.X }/database/
mysql/schema.sql
```

需要注意的是, 如果是安装 Proxy, 只导入 schema.sql 即可, 无须导入下面的 SQL, 否则 Proxy 无法正常工作。

```
shell# mysql -uzabbix -pzabbix zabbix < ${zabbix-X.X.X }/database/
mysql/images.sql
shell# mysql -uzabbix -pzabbix zabbix < ${zabbix-X.X.X }/database/
mysql/data.sql
```

确保以上过程无误。

```
shell# mkdir /var/log/zabbix
shell# chown zabbix.zabbix /var/log/zabbix
```

5. 复制 Service 启动脚本

```
shell# cp misc/init.d/fedora/core/zabbix_* /etc/init.d/
shell# chmod 755 /etc/init.d/zabbix_*
shell# sed -i "s#BASEDIR=/usr/local#BASEDIR=/usr/#g" /etc/init.d/
zabbix_server
shell# sed -i "s#BASEDIR=/usr/local#BASEDIR=/usr/#g" /etc/init.d/
zabbix_agentd
```

6. 配置 zabbix_server.conf 服务器端的文件

路径: /etc/zabbix/zabbix_server.conf。

修改主要参数即可正常工作。

DBName=zabbix 数据库名称

DBUser=zabbix 数据库用户

DBPassword=zabbix 数据库密码

```
shell# egrep -v "(#|^$)" /etc/zabbix/zabbix_server.conf #见第3章
```

7. 复制网页文件到 apache 目录

```
shell# cp -r ./zabbix-X.X.X/frontends/php/ /var/www/html/zabbix
shell# chown -R apache.apache /var/www/html/zabbix
```

开启 Zabbix 服务。

```
shell# chkconfig zabbix_server on
shell# chkconfig httpd on
shell# service zabbix_server start
shell# service httpd start
#如果启动失败, 请检查配置文件是否都正确
```

至此, Zabbix 的 Server 端安装完成。

8. 配置 Zabbix-Server 前端 UI

打开浏览器, 访问 <http://X.X.X.X/zabbix>, Web 配置的知识见第 3 章。

A.2 安装 Zabbix-Agent

安装 zabbix-Agent 的语句如下。

```
shell# wget http://sourceforge.net/projects/zabbix/files/ZABBIX%
20Latest%20Stable/2.2.2/zabbix-2.2.2.tar.gz
shell# groupadd zabbix -g 201
shell# useradd -g zabbix -u 201 -m zabbix
shell# tar xf ${zabbix-X.X.X}.tar.gz
shell# cd ${zabbix-X.X.X}
shell# ./configure --prefix=/usr --sysconfdir=/etc//zabbix --en
able-agent
shell# make
shell# make install
shell# mkdir /var/log/zabbix
shell# chown zabbix.zabbix /var/log/zabbix
shell# cp misc/init.d/fedora/core/zabbix_agentd /etc/init.d/
shell# chmod 755 /etc/init.d/zabbix_agentd
shell# sed -i "s#BASEDIR=/usr/local#BASEDIR=/usr/#g" /etc/init.d/
zabbix_agentd
shell# vim /etc/services
```

zabbix-agent	10050/tcp	#Zabbix Agent
zabbix-agent	10050/udp	#Zabbix Agent
zabbix-trapper	10051/tcp	#Zabbix Trapper
zabbix-trapper	10051/udp	#Zabbix Trappe

```
shell# sed -i "s/Server\=127.0.0.1/Server\=127.0.0.1,X.X.X.X/g"
/etc/zabbix/zabbix_agentd.conf #被动模式
shell# sed -i "s/ServerActive\=127.0.0.1/ServerActive\=X.X.X.X:
10051/g" /etc/zabbix/zabbix_agentd.conf #主动模式
```

注意，这里的 X.X.X.X 为 Zabbix-Server 的 IP 地址。

```
shell# sed -i "s#tmp/zabbix_agentd.log#var/log/zabbix/zabbix_ag
entd.log#g" /etc/zabbix/zabbix_agentd.conf
shell# sed -i "#UnsafeUserParameters=0#aUnsafeUserParameters=1\
n" /etc/zabbix/zabbix_agentd.conf
```

启动 Zabbix-Agent 服务的语句如下。

```
shell# chkconfig zabbix_agentd on
shell# service zabbix_agentd start
```

A.3 关于 Zabbix 的升级

从 Zabbix 1.8 升级到 Zabbix 2.0，需要对数据库打补丁，从 Zabbix 2.0 升级到 Zabbix 2.2 是自动对数据库进行升级的，步骤如下。

- ① 先对数据库进行备份，参考 3.12 节的数据库备份脚本。
- ② 升级 zabbix_server 二进制文件。

源码在 src/libs/zbxdbupgrade/dbupgrade.c 中，读者可以参考源码去研究其实现细节，如图 A-1 所示为部分数据库自动升级的代码。

```
static int DBpatch_2010034(void)
{
    return DBdrop_field("events", "value_changed");
}

static int DBpatch_2010035(void)
{
    const char *sql = "delete from profiles where idx='web.events.filter.showUnknown'";
    if (ZBX_DB_OK <= DBexecute("ms", sql))
        return SUCCEED;
    return FAIL;
}

static int DBpatch_2010036(void)
{
    const char *sql =
        "update profiles"
        " set value_int=case when value_str='1' then 1 else 0 end,"
        " value_str='', "
        " type=2" /* PROFILE_TYPE_INT */
        " where idx like '%isnow'";
    if (ZBX_DB_OK <= DBexecute("ms", sql))
        return SUCCEED;
    return FAIL;
}

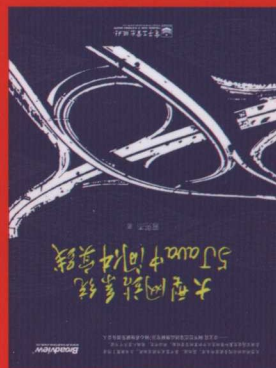
static int DBpatch_2010037(void)
{
    if (ZBX_DB_OK <= DBexecute("update config set server_check_interval=18"))
        return SUCCEED;
    return FAIL;
}

/*src/libs/zbxdbupgrade/dbupgrade.c" 2774L, 69009C written
```

图 A-1

关于 Zabbix 1.8、Zabbix 2.0、Zabbix 2.2 三个版本的数据库字段变更，请参考如下网址中的内容：

<https://github.com/itnihao/zabbix-book/blob/master/17-chapter/zabbix-database-1.8-2.0-2.2.xlsx>



【业内力荐】

运维离不开监控，就像鱼离不开水，一款功能强大的监控系统可以有力地保证业务性能的稳定。近几年各种监控系统层出不穷，Zabbix作为监控系统的新兴贵族迅速崛起，不过有关它的中文资料还比较少，itnihao作为Zabbix监控系统国内领先的使用者，厚积薄发撰写了此书，大家可以通过这本书详细了解Zabbix的各方面。

——窦喆 (@南非蜘蛛) 中国最大开源社区ChinaUnix创始人之一

对于略具规模的IT系统而言，监控组件通常都是其运维管理工具箱中核心组件的排头兵。在开源运动硕果累累的今天，监控工具领域也是百花齐放，然而，Zabbix作为后来者，却迅速成为一枝独秀，甚至由于愈加完善的特性和丰富的功能以及越来越多企业或组织的青睐而冠盖群芳。相信不少初次使用Zabbix的人都是从itnihao公开的且不断丰富和完善的Zabbix技术文档开始的，许多网友也坦言是itnihao和他的文档伴随自己走过了Zabbix资料匮乏、使用经验缺少的“蛮荒”时代。由此，我们有理由相信，他这次把多年的研究成果及实践经验精心打造并集结成册的这本《Zabbix企业级分布式监控系统》，一定会成为Zabbix学习者的案头必备宝典。

——马永亮 (@马哥教育) 马哥Linux运维培训创始人

本书是作者在运维领域多年实践的精彩总结，没有花哨的语言，不是大而全的砖头书。从头至尾，循序渐进，抽丝剥茧，会让你对Zabbix的基本操作及其原理有全面、系统的认识。作者大部分篇幅都采用图例加脚本实例阐述，相信这也是他对Zabbix架构的深刻理解，只有掌握了扎实的原理架构，才能让实战操作井井有条，避免出现一些人为的低级错误。本书一定能给Zabbix学习者带来帮助。

——黄小路 (@ZERO__0) PPS高级运维工程师

作为开源监控系统的一员，Zabbix提供了All in One的解决方案，使用户能够快速构建出企业级的监控平台，让运维环境变得可知可控。本书作为第一本中文Zabbix书籍，非常系统地讲解了Zabbix的各方面，从功能到部署和使用，从原理到案例技巧，几乎所有关于Zabbix的问题都可以从中找到答案或启发，值得一读！

——姚炫伟 (@绿小小肥) 中国SaltStack用户组发起人之一

本书理论与实践相结合，包含Zabbix的各项技术细节，由浅入深。无论是新手还是老用户，看完本书后，一定能给你带来技术和思路的拓展，正如作者所言，本书学的不止是技术，而是解决问题的思路。

——邝玲 高级系统运维工程师



博文视点Broadview



@博文视点Broadview



责任编辑：李利健

封面设计：李玲

上架建议：计算机 / 监控运维

ISBN 978-7-121-23877-2



9 787121 238772 >

定价：59.00元